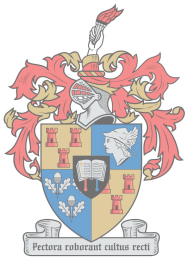
Recommender Systems with Bayesian Aspect Models and the Effect of Approximate Inference

by

Dylan Verrezen



Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. J.A. du Preez

March 2018

Abstract

Recommender Systems with Bayesian Aspect Models and the Effect of Approximate Inference

D. Verrezen

Department of Electrical Engineering,
University of Stellenbosch,
Private Bag X1, Matieland, 7602.

Thesis: MScEng

March 2018

Recommender systems form an important part of the modern world. These systems allow users to find relevant items in often huge item collections. Collaborative filtering is a pervasive and popular form of recommender that recommends to users based on their histories and the histories of other users. The field was long dominated by two forms of collaborative filtering: neighbourhood methods and matrix factorisation models. The two approaches were based on the assumption that the better the prediction of the rating a user would give an item, the greater the quality of the recommendations. This assumption has been criticised as being misleading. One major criticism is that the recommender systems are not being evaluated on the quality of the actual recommendation list. The systems are instead being evaluated on how well they perform a proxy task: predicting ratings. Another criticism is that it leads to recommenders that overfit to popular items that have the majority of the observed feedback. One possible improvement is to instead evaluate recommender systems by how effectively they rank items by determining how many relevant items they can return for a user in their top N results. However,

traditional, popular forms of collaborative filtering perform these tasks poorly. To address this we turn to Bayesian aspect models. These models come from the field of topic modelling. These aspect models are unsupervised models that express co-occurrence data in terms of latent aspects, where aspects are collections of thematically related items. The best known and most powerful aspect model is Latent Dirichlet Allocation, and preliminary literature suggests that it performs very well for recommendation tasks. A drawback to these Bayesian aspect models is that exact inference is intractable and we need to turn to approximate inference techniques. In this document we verify the performance of Latent Dirichlet Allocation for recommendation, and investigate the effect of approximate performance on the results for recommendation tasks.

Uittreksel

Recommender Systems with Bayesian Aspect Models and the Effect of Approximate Inference

D. Verrezen

Department of Electrical Engineering,
University of Stellenbosch,
Private Bag X1, Matieland, 7602.

Tesis: MScEng

Maart 2018

Aanbevelingstelsels speel 'n belangrike rol in die moderne wêreld. Hierdie stelsels stel gebruikers in staat om relevante items op te spoor in (dikwels) groot versamelings data wat oor verskeie domeine kan strek. Samewerkende filtrering maak gebruik van die soortgelykhede tussen 'n gebruiker se geskiedenis en dié van ander gebruikers om aanbevelings te maak. Hierdie veld was lank gedomineer deur twee vorme van samewerkende filters: buurtmodelle en matriksfaktorisering. Die twee benaderings was gegrond op die veronderstelling dat 'n beter voorspelling van die telling wat 'n gebruiker aan 'n item sou gee, noodwendig ook sal lei tot hoër gehalte aanbevelings. Hierdie aanname blyk misleidend te wees – die tellings wat 'n gebruiker gee is nie direk ekwivalent aan die nuttigheid van 'n aanbeveling nie. Verder lei dit ook tot 'n oormatige fokus op populêre items wat volop in die beskikbare data voorkom. Gevolglik het die veld verskuif na nuwe vorme van evaluering. 'n Moontlike benadering is om eerder aanbevelingstelsels te baseer op hoeveel van die items in die boonste N aanbevelings relevant vir die gebruiker was. Die tradisionele gewilde vorme van samewerkende filtrering vaar nie juis goed met hierdie tipe evaluering nie. Om

meer toepaslike stelsels te ontwikkel, wend ons ons tot Bayesiese aspekmodelle. Hierdie modelle is gewild in die veld van onderwerpsmodellering. Hierdie tipe van modelle kan sonder eksplisiete toesig assosiasies maak tussen items wat dikwels saam in tematies-verwante data voorkom. Die bekendste en sterkste aspekmodel is die sogenaamde latente Dirichlet-toekenningtegniek. Voorlopige ondersoeke dui daarop dat dit belofte vir aanbevelingstake mag inhou. 'n Nadeel van hierdie Bayesiese modelle is dat presiese inferensie wiskundig onhaalbaar is – dit noop mens om dit met benaderingstegnieke te takel. In hierdie dokument verifieer ons die nuttigheid van latente Dirichlet-toekenning vir aanbeveling, en ons ondersoek ook die rol wat benaderingstegnieke ten opsigte van aanbevelingstake speel.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2018 .....

Contents

List of Figures

List of Algorithms

Nomenclature

General Recommendation

$\boldsymbol{R}$	Feedback Matrix
U	Number of users
I	Number of items
u	A single user
i	A single item
$r_{u,i}$	Explicit feedback user u gave item i
$y_{u,i}$	Implicit feedback user u gave item i
$\hat{r}_{u,i}$	Predicted feedback user u gave item i
μ	Mean
σ	Standard deviation

Matrix factorisation

K	Number of features
$\boldsymbol{V}$	User feature matrix for all users
$\boldsymbol{Y}$	Item feature matrix for all items
$\hat{\boldsymbol{R}}$	Predicted feedback matrix

Latent Dirichlet Allocation

K	Number of aspects
z	Aspect membership indicator
$\boldsymbol{\beta}$	Aspect-item distribution. A $K \times I$ random matrix
$\boldsymbol{\beta}_k$	Item distribution for aspect k

$\boldsymbol{\eta}$ I-dimensional Dirichlet parameter for the prior on $\boldsymbol{\beta}_k$

$\boldsymbol{\theta}_u$ User u's aspect mixture. A K dimensional multinomial

$\boldsymbol{\alpha}$ K-dimensional Dirichlet parameter for the prior on $\boldsymbol{\theta}_u$

Approximate Inference

$x^{(t)}$ The state of the random variable x at time t

π The stationary distribution of a Markov chain

$\mathcal{Q}$ A proposal distribution

$\boldsymbol{x}_{-i}$ All of the elements in the vector $\boldsymbol{x}$, except the i-th element

Gibbs Sampling for Latent Dirichlet Allocation

$N_{-n,k}$ Number of items assigned to aspect k excluding the current item

$N^i_{-n,k}$ Number of times item i has been assigned to aspect k excluding the current item

$N^u_{-n,k}$ Number of items assigned to aspect k for user u excluding the current item

N^u_{-n} Number of items observed for user u excluding the current item

Variational Inference for Latent Dirichlet Allocation

$\boldsymbol{\lambda}_k$ Variational Dirichlet parameter on the variational approximation to $\boldsymbol{\beta}_k$

$\boldsymbol{\gamma}_u$ Variational Dirichlet parameter on the variational approximation to $\boldsymbol{\gamma}_u$

$\boldsymbol{\phi}_{u,n}$ Variational Multinomial parameter on the variational approximation to $z_{u,n}$

Mathematics

$exp(x)$ The natural exponent of x, (e^x)

$\mathbf{I}_N$ The identity matrix. A matrix with 1 for all diagonal values and 0 otherwise with rank N

$\mathbf{1}$ A vector of all ones

$\mathrm{KL}\left(p \,||\, q\right)$ Kullback-Leibler divergence between p and q

Distributions

$\mathrm{Dir}(\boldsymbol{\alpha})$ Dirichlet with parameter $\boldsymbol{\alpha}$

$\mathrm{Multi}(\boldsymbol{p})$ Multinomial with probabilities $\boldsymbol{p}$

Abbreviations

RV Random Variable

PDF Probability Density Function

PMF Probability Mass Function

CDF Cumulative Distribution Function

MLE Maximum Likelihood Estimation

MAP Maximum a Posteriori

SVD Singular Value Decomposition

PLSA Probabilistic Latent Semantic Analysis

LDA Latent Dirichlet Allocation

GAM Generative Aspect Model

MCMC Markov Chain Monte Carlo

KL Kullback-Leibler

VI Variational Inference

ELBO Evidence Lower Bound

CAVI Co-ordinate Ascent Variational Inference

SVI Stochastic Variational Inference

ADVI Automatic Differentiation Variational Inference

ADF Assumed Density Filtering

EP Expectation Propagation

LBP Loopy Belief Propagation

Chapter 1

Introduction

1.1 Motivation

Recommender systems are an important part of modern life. The sheer volume of content exposed by the internet requires systems that enable users to filter out relevant items from huge online collections. The basic task of a recommender system is to generate a list of relevant items for a user. This list could be movies that a user will enjoy, or items that a user would purchase. Recommender systems form an important part of many commercial systems such as for recommending video media at Netflix (Gomez-Uribe & Hunt, 2015), items for purchase at Amazon (Linden *et al.*, 2003; Smith & Linden, 2017) or news articles for the New York Times (Spangher, 2015). Recommender systems have also been used for other diverse tasks such as helping researchers find scientific articles (Wang & Blei, 2011).

1.2 Background

Originating in the mid-90s, the earliest recommender systems were based on collaborative filtering (Resnick *et al.*, 1994; Shardanand & Maes, 1995; Hill *et al.*, 1995), where the recommendation was based on the behaviour of similar users. The dominant method for evaluating recommender systems was based on predicting the rating a user would give an item. These early systems were based on neighbourhood methods. Neighbourhood methods generated

recommendations for a user by finding a neighbourhood for a user that consisted of similar users. The recommender system could then predict the ratings based on the ratings that similar users gave items. These methods are known as memory-based recommenders, because they do not construct any explicit model (Ekstrand *et al.*, 2011). These approaches remain a core approach for recommendation today, as they are intuitive and simple to implement (Schafer *et al.*, 2007).

1.2.1 Ratings Prediction Systems

These early methods generated recommendations by first predicting the rating a user would give an item. Items could then be recommended based on the predicted rating. These predictions were evaluated with average error metrics such as Root Mean Square Error (RMSE) between the predicted and actual ratings. With ratings, users directly indicate their preference; such ratings data is known as explicit feedback. The choice to focus on evaluating recommender systems with explicit data was in part driven by the fact that explicit data was the only form of feedback available (Gomez-Uribe & Hunt, 2015). By considering explicit feedback as a measure of user preference, the assumption was that improving the accuracy would provide better recommendations.

The focus on ratings prediction culminated in the Netflix Prize competition. The Netflix Prize was a competition with a $1,000,000 prize that would be awarded to the team that could achieve a 10% improvement over Netflix's own recommender system (Bell & Koren, 2007). The task was ratings prediction and was measured via RMSE. The Netflix Prize led to the development and subsequent popularity of matrix factorisation models. These models are closely related to dimensionality-reduction techniques such as Principal Component Analysis and low-rank singular value decomposition approximations. The core idea behind matrix factorisation models is to map users and items to a common low-rank feature space. The low rank of the feature space acts to bottleneck the model so that it can find general features that describe the users and items. Rating prediction can be performed using the feature representations of the users and items (Koren *et al.*, 2009).

1.2.2 Matrix Factorisation Models

The first matrix factorisation models used singular value decomposition (SVD) to find the feature representations of users and items. The SVD-based matrix factorisation model is commonly referred to as PureSVD to differentiate from models that came after (Koren *et al.*, 2009). A problem for PureSVD is that recommendation datasets are extremely sparse, i.e. most of the user-item feedbacks are unobserved. The singular value decomposition is not defined for sparse datasets and the workarounds to this are computationally expensive. A breakthrough for matrix factorisation models came in the form of an approximate method of calculating the SVD just using the observed data. The method uses stochastic gradient descent to learn the user and item features directly. The approximate SVD model led to increased interest in matrix factorisation models after it jumped to third place on the Netflix Prize leaderboards (Funk, 2006). This was despite its relative simplicity compared to the complex ensembles of models being used at the time.

Team Bellkor was a very successful team over the course of the Netflix competition and made heavy use of matrix factorisation models. They developed two extensions to the SVD based approaches, namely Asymmetric-SVD and SVD++ (Bell *et al.*, 2008). Team Bellkor ultimately won the Netflix Prize after joining up with another team, and the winning approach made heavy use of matrix factorisation models.

While the competition was a success and spurred the development of many new approaches to recommendation, the winning solutions involved complex ensembles of models that were impractical for real-world use (Gomez-Uribe & Hunt, 2015). However, Asymmetric-SVD and SVD++ remain state of the art for rating prediction, and matrix factorisation models became a core approach to recommendation.

1.2.3 Topic Models

Matrix factorisation models are not limited to recommender systems: early topic models also used matrix factorisation techniques. Topic modelling is a

field of information retrieval that attempts to find the latent semantic structure for collections of text (Blei & Lafferty, 2009). Topic models describe documents in terms of latent topics and aim to enable the indexing and discovery of documents in large document corpora. An early topic model was Latent Semantic Analysis (LSA), which used a low-rank SVD approximation to map documents to the topic space, much in the same way as PureSVD would for recommendation. Hofmann (1999) gives an equivalent statistical model to LSA in the form of Probabilistic Latent Semantic Analysis (PLSA). PLSA attempted to improve over LSA by finding better topics and to provide a statistical backing to explain why the matrix factorisation approach works.

PLSA tended to overfit and to address this Blei *et al.* (2003) extended PLSA to a Bayesian statistical model known as Latent Dirichlet Allocation (LDA). Latent Dirichlet Allocation has been very successful for topic modelling (Blei, 2012) and has been applied to many collections of documents, including newspaper articles (Wei & Croft, 2006) and scientific abstracts (Blei *et al.*, 2003).

Whereas topic models have been suggested for recommendation, they are ill-suited to rating prediction and failed to make much of an impact. When used for recommendation, topic models bypass predicting the ratings and instead directly predict items for a user. Hofmann (2004) develops PLSA for collaborative filtering and devises extensions to the model to enable ratings prediction. Blei *et al.* (2003) also suggests that LDA could be used for collaborative filtering.

1.2.4 Ratings Prediction versus Top-N Recommendation

The reason topic models cannot natively perform ratings prediction is that they were originally designed to work with word counts in documents. Recommendation also has a form of count data in the form of implicit feedback. Implicit feedback consists of observations of user behaviour that indirectly indicates user preference. This could be counts of how many times a user viewed an item or clicked a link. With the exploding popularity of the web, the amount

of implicit data saw a sharp increase. For example at Netflix their early focus on ratings data was based on the fact that when they were mailing DVDs to people, ratings were the only form of feedback they received (Gomez-Uribe *&* Hunt, 2015).

The need to work with implicit data and criticisms of ratings prediction as an evaluation method has led to alternative methods of evaluating recommender systems. Ratings prediction has been criticised because it measures recommendation performance based on the predicted rating and not on the quality of the recommendation list (Cremonesi *et al.*, 2010). Another criticism is that the focus on predicting the rating leads to models that overfit on popular items in the dataset (McNee *et al.*, 2006). Additionally, in practice the assumption that better ratings prediction led to higher quality recommendations was misleading (Gomez-Uribe *&* Hunt, 2015). An alternative method of evaluating recommender systems is instead to evaluate the accuracy of the recommendation list produced. One such method is to evaluate systems based on how many relevant items they can produce for a user in the system's top-N predictions.

Many of the state-of-the-art systems for recommendation perform poorly when evaluated via top-N prediction. Cremonesi *et al.* (2010) showed that simple early core approaches such as neighbourhood models and PureSVD matched and even exceeded the performance of state-of-the-art systems such as Asymmetric-SVD and SVD++. When performing evaluation via top-N prediction, the LDA model's inability to predict ratings is no longer a problem. Other evaluations of recommenders suggested that probabilistic approaches perform well for item prediction (Barbieri *&* Manco, 2011). With the shifting focus towards item prediction, we can revisit topic models like LDA for recommendation. This document aims to develop the LDA model in the context of recommendation and to show how it evolved from matrix factorisation models. It also aims to evaluate how LDA performs for item prediction compared to the core recommender system methods. One specific case of recommendation that is of particular interest is recommending to new users in a system. This is a challenging task, because a new user has little or no history from which to generate

recommendations. This is known as the cold-start problem. Bayesian models are known to gain accuracy quickly using little data (Ng & Jordan, 2002). Investigating how LDA works in a cold-start scenario is therefore of particular interest.

1.2.5 Bayesian Models and Approximate Inference

A drawback of Bayesian models is that for many interesting models including LDA, exact inference is intractable. When performing Bayesian inference we are interested in calculating the *posterior* distribution of the model. The posterior is the distribution of the model's variables given the observed data. Unfortunately, calculating the posterior involves integrals that are intractable.

This means we must turn to approximate inference algorithms to learn the model parameters. A popular approach to approximate inference is Markov Chain Monte Carlo (MCMC) methods. These methods leverage Monte Carlo integration and samples from a Markov chain to approximate the intractable integrals involved in inference (Gilks *et al.*, 1995). Other approaches involve finding a simpler approximate posterior that is close to the true posterior. Methods that do so rely on minimising some divergence measure between the approximate and true posterior. One such divergence measure is Kullback-Leibler divergence. Kullback-Leibler divergence is not symmetrical, meaning that for a posterior p and approximate posterior q the Kullback-Leibler divergence between $p||q$ is not the same as that between $q||p$. Two approaches that find approximate posteriors by minimising the Kullback-Leibler are Variational Inference (VI) (Blei *et al.*, 2017) and Expectation Propagation (EP) (Minka, 2001). VI and EP use the opposing symmetries of the Kullback-Leibler divergence as each other to find the approximate posterior. What is not clear is the effect that the assumptions, simplifications and objectives of these approximate inference algorithms will have on the LDA performance in practice. This document also seeks to investigate how the choice of approximate inference will affect the suitability of the LDA model for recommendation.

1.3 Goals

1. Develop a recommender system based on LDA.
2. Compare recommenders based on LDA against the core recommendation approaches.
3. Investigate the effect that approximate inference has on recommendation accuracy when using LDA for recommendation.
4. Investigate LDA for cold-start recommendation.

1.4 Contributions

1. Re-formulating a number of recommendation systems in a common framework, thereby making their interrelationships clear.
2. C++ code implementations of the following approximate inference algorithms for LDA:
   a) Variational inference.
   b) Collapsed Gibbs sampling.
   c) Expectation Propagation.
3. Python code implementations of the following recommendation approaches:
   a) Neighbourhood methods.
   b) The matrix factorisation model: PureSVD.
4. Experimental results performed on a dataset of one million ratings from the MovieLens data set:
   a) Showing that the LDA recommender consistently outperforms traditional approaches for top-N recommendation.
   b) Showing that the LDA recommender gains its advantage from improved personalisation.
   c) Showing that the LDA recommender has promising results for cold-start recommendation.

1.5 Overview

We are interested in evaluating the LDA topic model for recommendation. LDA is a Bayesian generative model. Bayesian statistics consider probability as a rational measure of uncertainty and treat all uncertain variables, including the model parameters, as random variables. Chapter 2 provides a review of some theoretical background for the Bayesian model. An important part of this is covering how the models are represented as a generative process (Section 2.1.3) coupled with a graphical model (Section 2.7). The graphical model representation helps to illustrate the dependency structure of the model (Section 2.7). This allows us to factorise the model's joint distribution into a product of conditional distributions for the model parameters. This representation is also useful as many of the inference techniques in Chapter 5 require a factorised representation of the model.

Before developing LDA for recommendation, we cover it precursor models. Many state-of-the-art collaborative filtering recommender systems are based on matrix factorisation.

1.5.1 Recommender Systems

Recommender systems are tasked with discovering new items for users that they will enjoy. A common representation for recommender systems is the feedback matrix:

$$\boldsymbol{R}_{U\times I} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,I} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,I} \\ \vdots & \vdots & \ddots & \vdots \\ r_{U,1} & r_{U,2} & \cdots & r_{U,I} \end{pmatrix}. \tag{1.1}$$

The feedback matrix is a matrix of all the implicit or explicit feedback observed from users, placed into spares matrices with users as rows and items in columns. Many early recommender systems were evaluated on how well they could predict the missing values of the matrix. More recently, evaluation of recommender systems has started instead to move towards evaluating the sys-

tem on how well it can predict the items a user would choose. More detailed coverage of the recommendation problem is given in Section 3.1.

Recommendation accuracy is not the only factor when evaluating recommender systems. There are many other important aspects that should be considered when designing or evaluating a recommender. One of the most important is *explainability*, whereby the recommendations provided by the system can be explained to the user. Users generally prefer recommendations where they understand why items are being recommended to them. Matrix factorisation methods often cannot generate explainable recommendations.

Matrix factorisation methods for recommendation are covered in Section 3.5. The core idea of these methods is to map users and items to a latent feature space that is of a lower dimension that the feedback matrix. To do so, the simplest methods factorise the feedback matrix into lower-ranked feature matrices, namely a user- and an item-feature matrix. Ratings can be predicted by the dot product of a user and item in feature space. Recommendations are then performed by ranking items by their predicted ratings.

Probabilistic Latent Semantic Analysis (PLSA) is a statistical interpretation of matrix factorisation models. PLSA restricts the feature space to be a valid probability distribution. In doing so, items are characterised by their membership to aspects. An aspect is a distribution over all the items and captures the notion of a user community, or items that frequently occur together for certain users. The users are characterised by their membership of aspects and they can proportionally belong to several aspects. This characterisation captures how users belong to the pseudo-communities or how they can have diverse interests. PLSA suffered from drawbacks such as overfitting, much like matrix factorisation models, and it did not capture all the available relationships between items.

LDA further extends PLSA to be fully Bayesian. This addresses some of the overfitting concerns and allows for the use of the tools from Bayesian modelling, such as easy model extensions. In theory LDA is a good model for

recommendations - it directly estimates the probability of users choosing items and it naturally partitions users into communities and items into clusters. It is a standard model for topic modelling; however, it is not well understood for recommendation.

1.5.2 Approximate Inference

A hurdle to working with Bayesian models is that all models must define a valid joint probability distribution over the observed data and the model parameters. For most interesting models this involves intractable integrals. We can instead turn to approximate methods that estimate the posterior distribution of the model.

One possible approach to approximate inference is to use sampling methods as shown in Section 5.2. Markov Chain Monte Carlo (MCMC) is a sampling method that constructs a Markov chain to draw samples from the posterior distribution of the model. It then uses Monte Carlo integration with the generated samples to approximate the intractable integrals with sums.

The second class of approximate inference method reframes inference as an optimisation problem. By defining an approximate posterior, the goal becomes to optimise the approximation to be as close as possible to the true posterior. To do so, such methods minimise a divergence measure between the approximate and true posterior. A common divergence measure used for this is Kullback-Leibler divergence; however, Kullback-Leibler divergence is not symmetrical. The choice of whether to minimise between the true and approximate or vice-versa changes the nature of the approximate posterior found. More discussion on Kullback-Leibler divergence can be found in Section 5.3.

Variational Inference and Expectation Propagation are two such approaches that minimise Kullback-Leibler divergence. Both approaches will approximate a multi-modal distribution with a simpler distribution with some key difference. Variational Inference is mode-seeking, and will find an approximate distribution that closely matches one of the true distribution's modes. In

contrast, Expectation Propagation is moment-matching, and will find an approximate distribution that averages over the modes of the true distribution. The found approximation has moments that match the moments of the true distribution. Variation Inference is discussed in Section 5.4 and Expectation Propagation in secion 5.5.

1.5.3 Key Results and Conclusions

The goal of the experiments was to investigate how well LDA performs for recommendation in comparison to the core recommender approaches, and to evaluate how much the choice of approximate inference technique affects the performance of LDA. Our first experiment compared Latent Dirichlet Allocation to the best-performing approaches from Cremonesi *et al.* (2010). LDA utilising variational inference consistently outperformed the other approaches. The effect of inference was noticeable and the LDA had similar performance to the other approaches when not using variational inference.

The second experiment dug deeper into LDA's strong results in an attempt to uncover whether the accuracy arose from improved personalisation of the recommendations. To investigate the effect of personalisation, recommendations were made from the same trained LDA model via two methods. The first was the standard personalised approach, while the second ignored the user features and generated recommendations directly from the global item features. When unpersonalised, the LDA model performs similarly to the other recommendation methods. This indicates that most of the improvement came from better tailoring the recommendations to the users and not from finding better general recommendations.

The final experiment explored LDA in cold-start scenarios. A cold start in recommendation is when the system has to recommend to users with little or no history, e.g. new users. For this experiment only a small fraction of the test set user ratings were used for training. The LDA still performed well, but not drastically better as it did in the regular case.

Overall Latent Dirichlet Allocation showed very promising performance when used for recommendation. The effect of approximate inference cannot be ignored, however. Variational inference made the difference between LDA performing roughly as well as the other approaches, or performing nearly twice as well. A further advantage for LDA is that it is a popular and well-studied model for topic modelling. Many extensions that may improve the model for recommendation already exist. Further investigation of LDA for recommendation could consider how these existing extensions could enable recommendation. Alternatively, LDA's Bayesian nature allows for it to be easily adapted to new recommender domains or to find novel extensions specifically for recommendation.

Chapter 2

Bayesian Modelling

Statistical modelling forms an important part of the scientific process and for machine learning. We will cover the model Latent Dirichlet Allocation, which is a Bayesian model over co-occurrence data. This chapter gives some brief theoretical background on Bayesian modelling and how the models are represented; Latent Dirichlet Allocation is covered in the next chapter.

2.1 Statistical Modelling

Statistical modelling is concerned with modelling some data-generating process where some or all of the variables involved are stochastic. When a variable is stochastic we refer to it as a random variable (RV). A random variable is the numerical outcome of a random process and does not have a fixed deterministic value, but we instead describe it via a probability distribution (Peebles & Shi, 2001). The probability distribution assigns a probability (or a probability density in the case of continuous RVs) to every possible value of the random variable. Probability itself is generally interpreted in one of two ways: frequentist or Bayesian (Bishop, 2006). The two different perspectives on probability lead to different approaches in modelling.

2.1.1 Frequentist and Bayesian Paradigms

The frequentist or classical viewpoint denotes probability as the relative frequency of a random and repeatable event. Frequentist modelling regards the

observed data as random and unreliable and seeks to estimate the true value of the model parameters (Bishop, 2006).

By contrast, the Bayesian viewpoint is a more general view of statistics that considers probability as a rational, conditional measure of belief (Bishop, 2006). The Bayesian paradigm is axiomatic and all methods only require the interpretation of probability as a measure of uncertainty and the mathematics of probability theory. It has been shown that in terms of the Bayesian viewpoint, probability theory is an extension of Boolean logic with uncertainty (Jaynes, 2003). A central component of this approach is that all unknown quantities are described by probability distributions. This includes the model parameters. The probability distributions over the variables quantify our uncertainty about the variables and allow us to revise these beliefs with new evidence. A characteristic of the Bayesian approach is that *prior* distributions must be supplied that describe the initial beliefs about variables and parameters. This allows for prior knowledge to be naturally included in models (Bishop, 2006).

2.1.2 Generative Models

Statistical models specify a model that is assumed to describe the true process generating the observable data, hence the name *generative models*. A characteristic of generative models is that they can be sampled to produce synthetic data. Modelling involves many simplifying assumptions and the true distribution of the data is probably far too complex to precisely capture, but models can still serve as useful approximations to reality (Burnham *&* Anderson, 2003). Recommender systems typically have extremely sparse input observations, and generative models have been shown to work well with few observations (Ng *&* Jordan, 2002).

2.1.3 Generative Process

To specify a generative model, we can specify a pseudo-code-style process that specifies how each variable is generated. For a simple model with a single

distribution over the vector observations $\boldsymbol{x}$ parametrised by a parameter ζ, the basic generative process would be:

1. Sample the model parameters from the prior $\zeta \sim \mathrm{p}(\zeta)$.
2. Select (sample) each observation x in $\boldsymbol{x}$ from the data-generating distribution: $x \sim \mathrm{p}(x \mid \zeta)$.

2.2 The Rules of Probability

Bayesian statistics uses the mathematics of probability theory to represent uncertainty. The two most important rules of probability for Bayesian methods are the sum and the product rule.

2.2.1 The Sum Rule

For two discrete RVs x and y with a joint discrete probability distribution $\mathrm{p}(x, y)$, the *sum rule* allows us to get the *marginal* probability mass function of x:

$$\mathrm{p}(x) = \sum_{y} \mathrm{p}(x, y). \tag{2.1}$$

Equivalently, if x and y are continuous, the marginal probability density of x is found by integrating out y:

$$\mathrm{p}(x) = \int \mathrm{p}(x, y) dy. \tag{2.2}$$

2.2.2 The Product Rule

The second rule is the *product rule*. The product rule gives the relationship between the joint distribution and the conditional probability, $\mathrm{p}(y \mid x)$:

$$\mathrm{p}(x, y) = \mathrm{p}(y \mid x)\,\mathrm{p}(x) = \mathrm{p}(x \mid y)\,\mathrm{p}(y). \tag{2.3}$$

Using the symmetry property for the joint distribution

$$\mathrm{p}(x, y) = \mathrm{p}(y, x) \tag{2.4}$$

and by applying the product rule twice, we can obtain the *inverse probability* or as it is more commonly known, *Bayes' Theorem*:

$$\begin{aligned} \mathrm{p}(y \mid x) &= \frac{\mathrm{p}(x \mid y)\,\mathrm{p}(y)}{\int \mathrm{p}(x \mid y)\,\mathrm{p}(y)dy} \\ &= \frac{\mathrm{p}(x \mid y)\,\mathrm{p}(y)}{\int \mathrm{p}(x, y)dy} \\ &= \frac{\mathrm{p}(x \mid y)\,\mathrm{p}(y)}{\mathrm{p}(x)}. \end{aligned} \tag{2.5}$$

2.3 Bayesian Models

Consider the general case where we have observed some data $\mathcal{D}$ and we wish to learn something about the process that generated the observations. To do so we construct a generative model with parameters $\boldsymbol{\zeta}$ over the variables we have measured. Under the Bayesian paradigm the observations and model parameters $\boldsymbol{\zeta}$ are considered as random variables and the model defines a joint distribution over them $\mathrm{p}(\mathcal{D}, \boldsymbol{\zeta})$. The key problem is then finding an appropriate choice of model parameters, or using the Bayesian perspective quantify the uncertainty about their values. Specifically we are interested in the probability distribution over the model parameters given the observed data. This is known as the *posterior* probability:

$$\text{posterior} = \mathrm{p}(\boldsymbol{\zeta} \mid \mathcal{D}). \tag{2.6}$$

The posterior probability can be represented explicitly using Bayes' theorem (equation 2.5) in the form (Bernardo, 2003):

$$\mathrm{p}(\boldsymbol{\zeta} \mid \mathcal{D}) = \frac{\mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{\zeta})}{\mathrm{p}(\mathcal{D})}. \tag{2.7}$$

This requires first specifying a *prior* distribution $\mathrm{p}(\boldsymbol{\zeta})$ over the model parameters, capturing the assumptions or the initial beliefs about the parameters. The data affects the posterior through its influence in the *likelihood* function $\mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})$ in the numerator. The likelihood function is a function of the model parameters and evaluates how probable the observations are for the different values that the parameters could take. The likelihood function is not a valid probability distribution and does not necessarily integrate to one (Bishop,

2006).

The denominator $\mathrm{p}(\mathcal{D})$ is known as the *evidence* or *marginal likelihood*. It ensures that the posterior distribution is a valid distribution that integrates to unity (or sums if the posterior is discrete).

$$\begin{gathered} \mathrm{p}(\mathcal{D}) = \sum_{\boldsymbol{\zeta}} \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta}) \\ \text{or} \\ \mathrm{p}(\mathcal{D}) = \int \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}. \end{gathered} \tag{2.8}$$

The marginal likelihood is often omitted to give Bayes' rule in proportional form:

$$\begin{gathered} \text{posterior} \propto \text{likelihood} \times \text{prior} \\ \mathrm{p}(\boldsymbol{\zeta} \mid \mathcal{D}) \propto \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta}). \end{gathered} \tag{2.9}$$

The Bayesian approach is inherently iterative, and as new observations come in, the previous posterior can be used as a prior (both are distributions over the model parameters) in Bayes' rule to get an updated posterior with the new data. The model can update its beliefs (the posterior) as evidence comes in while taking into account its current beliefs (the prior).

2.3.1 Hierarchical Models

For models that do not just consist of a single distribution for the likelihood and prior, a more efficient representation is needed. In hierarchical models the statistical model is represented as a series of sub-models (Allenby *et al.*, 2005). Essentially the model consists of multiple levels of priors (Rouder *et al.*, 2013).

A simple two-stage hierarchical model which has a likelihood:

$$\mathrm{p}(\boldsymbol{x} \mid \boldsymbol{\zeta}) \tag{2.10}$$

a prior on $\boldsymbol{\zeta}$ with hyperparameter ϕ:

$$\mathrm{p}(\boldsymbol{\zeta} \mid \phi) \tag{2.11}$$

and the hyperprior on ϕ:

$$\mathrm{p}(\phi) \tag{2.12}$$

combining to get the posterior:

$$\mathrm{p}(\boldsymbol{\zeta}, \phi \mid \boldsymbol{x}) \propto \mathrm{p}(\boldsymbol{x} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta} \mid \phi)\, \mathrm{p}(\phi) \tag{2.13}$$

2.3.2 Conditionally Conjugate Models

The specific class of hierarchical Bayesian models that Latent Dirichlet Allocation belongs to are *conditionally conjugate* models. These models have levels of variables, split into local and global levels. Typically each observation is accompanied by a local latent variable. The global variables control the distribution over the latent variables.

2.4 Applying Bayes' Rule

To illustrate how Bayes' rule updates beliefs, we consider the simple example of trying to determine if a fair looking coin is biased. In the extreme case, if the coin is flipped a few times and comes up the same tails each time, the maximum likelihood estimate will suppose that the tails has a probability of 1, implying that all future flips will come up tails. With an appropriately chosen prior, the Bayesian approach can help avoid this overfitting but at the cost of having to choose a sensible prior distribution.

To do so, we consider a simple model with the bias of a coin θ as the probability it comes up heads. A normal fair coin will have a bias of $\theta = 0.5$ where heads and tails are equally likely, but we consider the case where we suspect the coin is biased toward one face, perhaps to give an edge in a game of chance.

The generative process for the biased coin example is given:

1. Select a bias by sampling from a symmetrical Beta distribution: $\theta \sim \mathrm{Beta}(\alpha, \beta)$ with equal parameters $\alpha = \beta$
2. For each coin toss, the result x_n is chosen as heads with probability θ and tails $1 - \theta$: $x_n \sim \mathrm{p}(\theta)$

We represent our prior beliefs about the coin as a symmetrical Beta distribution with equal hyperparameters so that it is centred on 0.5. The PDF for the Beta distribution is given in equation 2.14 where the function $B(\alpha, \beta)$ is a normalising function.

$$\mathrm{p}(\theta \mid \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} \tag{2.14}$$

If we flip the coin N times and observe k heads, our likelihood is given by the Binomial PMF where $\binom{N}{k}$ is the Binomial coefficient:

$$\mathrm{p}(k \mid \theta, N) = \binom{N}{k}\theta^{k}(1-\theta)^{N-k}. \tag{2.15}$$

We can now use Bayes' rule to update our belief about the bias θ with the observed k heads from N flips:

$$\mathrm{p}(\theta \mid k, \alpha, \beta) \propto \binom{N}{k}\theta^{k}(1-\theta)^{N-k}\frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} \tag{2.16}$$

Folding all normalising functions and constants into the implied normaliser in Bayes rule we get:

$$\begin{aligned} \mathrm{p}(\theta \mid k, \alpha, \beta, N) &\propto \theta^{k}(1-\theta)^{N-k}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ \mathrm{p}(\theta \mid k, \alpha, \beta, N) &\propto \theta^{\alpha+k-1}(1-\theta)^{\beta+N-k-1} \end{aligned} \tag{2.17}$$

Which, when we consider the form of the Beta distribution in equation 2.14 is a Beta distribution with parameters $\alpha + k$ and $\beta + N - k$.

$$\mathrm{p}(\theta \mid k, \alpha, \beta, N) = \mathrm{Beta}\left(\alpha + k, \beta + N - k\right). \tag{2.18}$$

In figure 2.1 the prior is shown with the posterior. We can see that after observing 8 heads out of 10 flips, the model believes that the coin may be biased. The belief is not extreme, however, as the data needs to overcome the prior. The prior in essence acts to regularise the model and allows us to represent our previous knowledge about it.

2.4.1 Maximum Likelihood and Maximum a Posteriori Estimation

In contrast to the Bayesian approach, the frequentist viewpoint considers the data to be uncertain or incomplete and wishes to estimate the *fixed* value

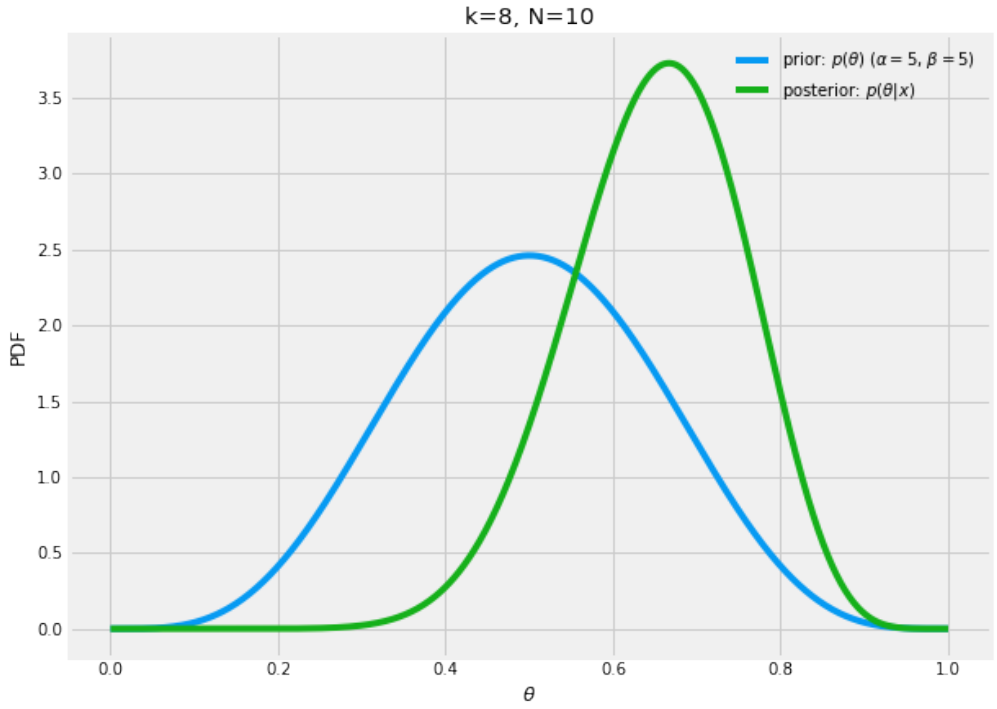


Figure 2.1: The prior Beta distribution (shown in blue) and the updated posterior distribution (shown in green) that we obtain after observing eight heads from ten coin flips. The two distributions are shown together to illustrate how the posterior represents our uncertainty about the coin. With the relatively low number of flips our updated belief suggests that the coin may be slightly biased but does not assume that the coin has a bias of 0.8 as a naive estimate would assume.

of the model parameters $\boldsymbol{\zeta}$. The uncertainty of this estimate can be shown with error bars obtained by considering the distribution of possible datasets (Bishop, 2006).

One possible method of obtaining estimates of the model parameters is via maximum likelihood estimation (MLE). MLE chooses the values for $\boldsymbol{\zeta}$ that best explain the observed data by finding the values for $\boldsymbol{\zeta}$ that maximise the likelihood:

$$\hat{\boldsymbol{\zeta}}_{MLE} = \arg\max_{\boldsymbol{\zeta}} \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta}). \tag{2.19}$$

Another method for obtaining a fixed-point estimate of the parameters that incorporates a prior is via *Maximum a Posteriori* (MAP) estimation. MAP finds settings for the parameter values by maximising the posterior as opposed to the likelihood. By exploiting the fact that the marginal likelihood does not depend on $\boldsymbol{\zeta}$ and is always positive, we can drop the denominator from Bayes' rule and get the MAP estimator as:

$$\hat{\boldsymbol{\zeta}}_{MAP} = \arg\max_{\boldsymbol{\zeta}} \mathrm{p}(\boldsymbol{\zeta} \mid \mathcal{D}) = \arg\max_{\boldsymbol{\zeta}} \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta}). \tag{2.20}$$

These approaches differ from the Bayesian perspective where $\boldsymbol{\zeta}$ is not considered as a fixed parameter but instead the uncertainty about $\boldsymbol{\zeta}$ is expressed for the given data in the posterior. A disadvantage to the Bayesian approach is that it requires calculating the marginal likelihood - the integral of which can, and often will for interesting models, become intractable to compute. In these cases we must turn to methods that approximate the posterior.

2.5 Choice of Priors

An important consideration for Bayesian modelling is selecting a representation for the prior beliefs.

2.5.1 Types of Priors

The Bayesian approach assumes the use of *subjective priors* where the priors are non-arbitrary and subjectively chosen to best represent our beliefs. The priors do not have to be overly specific and vague priors are useful since it directs the posterior toward useful models. One way of generating subjective priors is by soliciting them from experts, and they can be checked by generating data from them and comparing with our expectations of them.

For the case where we are ignorant of the nature of our priors we may use *objective priors*. These are non-informative priors that are chosen to minimise the impact of the prior.

2.5.2 Conjugate Priors

With Bayes' rule the normalisation constant requires that the product of the prior distribution and likelihood functions must be integrated over the parameter space (Fink, 1997). This integral may not always be analytically tractable. One of the methods for dealing with this is to derive pairs of likelihood functions and prior distributions with convenient mathematical properties such as tractable solutions to the integral (Fink, 1997).

We saw with the biased coin example that a binomial density for the likelihood and a Beta density on the prior yielded a Beta posterior. This kind of mathematical convenience is attractive when modelling as we have a well understood form for the posterior. In general for a likelihood function with an associated prior, the prior is conjugate to the likelihood when the posterior has the same form as the prior. When the prior is conjugate to the likelihood the posterior is mathematically tractable. When the distributions are not conjugate, the posterior may not be tractable depending on the form of the integral used when calculating the marginal likelihood.

2.5.3 Empirical Priors: Empirical Bayes

An alternative to directly specifying priors is to learn the prior parameters (hyperparameters) from the data. This avoids the problem of misspecifying the prior and misleading the model, but may overfit as data is double-counted.

2.6 Exchangeability

An important and common assumption for Bayesian modelling is *exchangeability*. Exchangeability captures the notion that only the value of observations matters and not the order in which they are observed. Exchangeability is an important dependency property for a sequence of random observations, and has notable consequences for the models that operate over them (Bernardo, 1996). In the field of recommender systems the most common representation of data assumes exchangeability, although it is not always stated.

For an observed sequence $\mathcal{D} = \{x_1, \ldots, x_n\}$ we model the joint probability with the density $\mathrm{p}(\mathcal{D}) = \mathrm{p}(x_1, ..., x_n)$. This joint density that we specify must capture the dependency amongst the individual x_i. The dependency structure can take many forms, but the exchangeability assumption is a simple one that is powerful and broadly applicable (Bernardo, 1996).

Exchangeability captures symmetry in $\mathcal{D}$ by specifying that the order of x_i's is uninformative in the sense that the information gained from $\mathcal{D}$ is independent of the order of collection for the individual x_i's (Bernardo, 1996).

Bernardo (1996) provides the formal definition: a sequence is considered *exchangeable* by requiring that

$$\mathrm{p}(x_1, ..., p_n) = \mathrm{p}(x_{\pi(1)}, \ldots, x_{\pi(n)}) \tag{2.21}$$

for all permutations π on the set $\{1, \ldots, n\}$ for every finite set of them.

Exchangeability has important consequences implied by the *general representation theorem* summarized by Bernardo (1996):

> The detailed mathematics of the representation theorems are involved, but their main message is very clear: if a sequence of observations is judged to be exchangeable, then any subset must be regarded as a random sample from some model, *and* there exists a prior distribution on the parameter of such model, hence requiring a *Bayesian* approach.

The general representation theorem is just an existence theorem; it tells us that any exchangeable sequence can be modelled by a Bayesian model but gives no insight into the nature of the model.

2.7 Graphical Models

Bayesian models represent the entire joint distribution which grows exponentially with the number of variables; Bayesian networks exploit the dependency structure of the joint to achieve a compact representation (Pearl & Russel,

2001). This compact representation allows for the joint distribution to be represented tractably even when it is extremely large Koller & Friedman (2009). The graphical form can be thought of as representing the set of dependencies, which in part allows for the factorization of the distribution (Koller & Friedman, 2009). We will represent all hierarchical models as a generative process accompanied by a Bayes network.

2.7.1 Directed Graphs: Bayesian Networks

Bayesian networks are directed acyclic graphs where nodes are random variables and the edges (which are directed) indicate dependencies amongst the variables (Pearl & Russel, 2001); observed variables are indicated by a shaded node (Clark & Thayer, 2004).

In figure 2.2 the Bayesian network for the coin-toss example is shown.

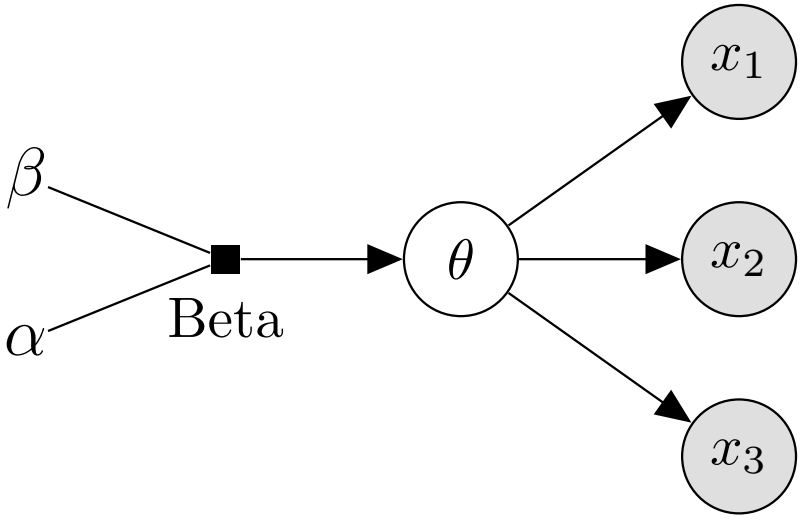


Figure 2.2: A Bayes network for the coin-toss experiment discussed previously, now with three flips. The coin flip results, $x_{1:3}$, are observed and indicated by the shaded nodes on the right of the diagram. The results depend on the coin bias, θ, which is not observed and is indicated by the middle unshaded node. The coin bias has a Beta prior with fixed parameters α and β. The prior parameters (hyperparameters) are not random variables and are thus not shown as a node.

Every Bayesian model needs to implicitly or explicitly represent the full

joint distribution of all variables. To understand why, consider the marginal likelihood in the denominator of Bayes' theorem for a Bayesian model with parameters $\boldsymbol{\zeta}$. Expand inside the integral with the marginal probability rule

$$\mathrm{p}(\mathcal{D}) = \int \mathrm{p}(\mathcal{D} \mid \boldsymbol{\zeta})\, \mathrm{p}(\boldsymbol{\zeta}) d\boldsymbol{\zeta} = \int \mathrm{p}(\mathcal{D}, \boldsymbol{\zeta}) d\boldsymbol{\zeta} \tag{2.22}$$

The problem with directly representing the joint is that it grows exponentially with the number of variables. For the simplest case where each variable is binary, with N variables the full joint distributions grow as a factor of $O(2^N)$. Using the product rule, the joint can be represented as the product of a set of complete conditionals. A complete conditional of a variable is the conditional distribution of that variable given all the other variables in the model. The Bayes network representation shows the dependency structure between the variables so that each complete conditional can be given only in terms of the parameters on which it depends.

For a Bayes Network over a set of variables $\boldsymbol{z}$, the joint is represented as the product of each variable's conditional distribution. The network represents the joint as a set of local conditional distributions of each node given its parents, allowing for the compact representation Pearl & Russel (2001):

$$\mathrm{p}(\boldsymbol{z}) = \prod_i \mathrm{p}(z_i \mid \mathit{parents}(z_i)) \tag{2.23}$$

If each variable node has no more than K parents, the complete network grows linearly with N as $O(N \cdot 2^K)$. As a simple example, consider the network below with five variables $\{z_1, ..., z_5\}$:

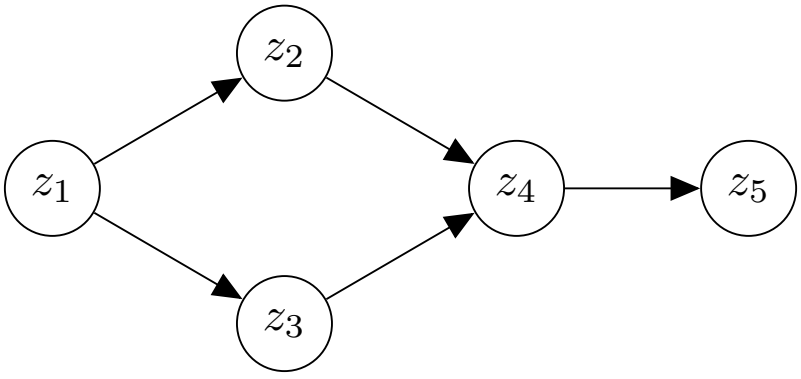


Figure 2.3: An example Bayes network with five random variables. The graph is a valid Bayes network as it is directed and acyclic.

The full joint is represented by the Bayes network as the following:

$$\mathrm{p}(z_1, z_2, z_3, z_4, z_5) = \mathrm{p}(z_1)\,\mathrm{p}(z_2 \mid z_1)\,\mathrm{p}(z_3 \mid z_1)\,\mathrm{p}(z_4 \mid z_2, z_3)\,\mathrm{p}(z_5 \mid z_4) \quad (2.24)$$

This compact representation is very convenient when performing many types of approximate inference that require a conditional factorisation of the joint.

2.7.2 Dependency Structure

The representation where each variable has a conditional distribution given its parents gives us the global dependency semantics of a Bayes network, which state that a node is conditionally independent of its non-descendants given its parents. This conditional independence is what allows for the compact representation of the Bayes network. However, we may wish to explicitly determine whether two variables (nodes) are conditionally independent when some of the nodes are observed (for which we have evidence). This leads to the concept of D-separation, which lets us determine if nodes are conditionally dependent. The 'd' in d-separation stands for directed and if two nodes are d-separated they are conditionally independent given the evidence.

Two nodes are considered d-separated if all paths between them are *inactive*; to understand an active path, consider the three possible triplets for the variables $\{a, b, c\}$ as seen in figure 2.4 and figure 2.5 (Charniak, 1991). A triplet is an active triplet given some evidence if a and c are not necessarily independent. This does not mean that they are definitely dependent, just that we cannot state that they are independent.

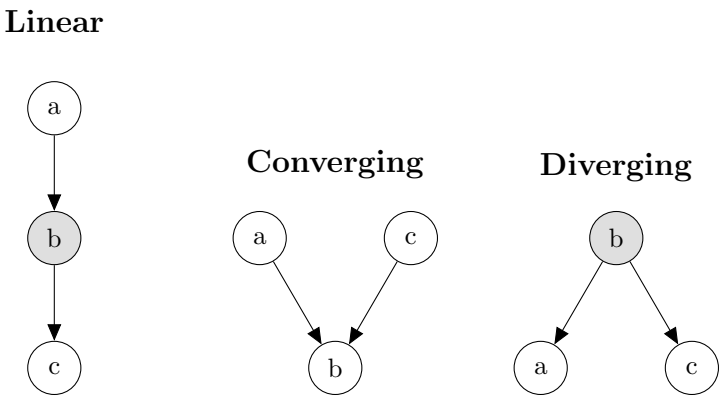


Figure 2.4: The three possible inactive triplet Bayes network configurations.

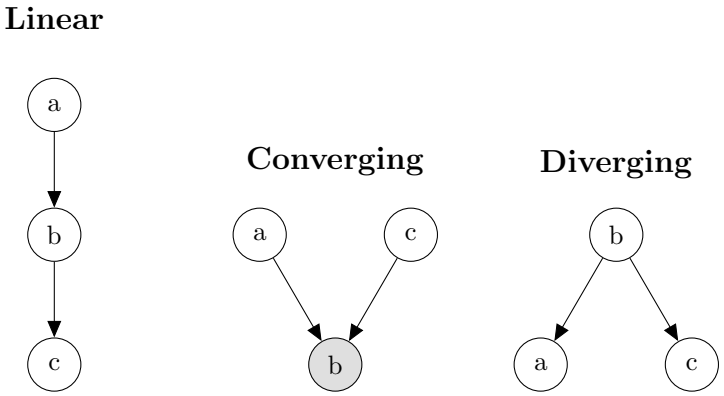


Figure 2.5: The three possible active triplet Bayes network configurations.

To understand when a is independent on c given b as evidence (observed) we can look at the three possible configurations. Recall that a and c are conditionally independent given b if

$$\begin{gathered} \mathrm{p}(a \mid b, c) = \mathrm{p}(a \mid b) \\ \text{or} \\ \mathrm{p}(c \mid a, b) = \mathrm{p}(c \mid b) \end{gathered} \tag{2.25}$$

To aid understanding it can be useful to visualise the directed edges as causality.

Linear

For the linear case the configuration can be seen as a *causal chain*. The Bayes network gives the joint as

$$\mathrm{p}(a, b, c) = \mathrm{p}(a)\,\mathrm{p}(b \mid a)\,\mathrm{p}(c \mid b) \tag{2.26}$$

If b is not observed, we do not know if a is independent of c, so the configuration is active. However, if b is observed, we can show that a is independent of c

$$\begin{aligned} \mathrm{p}(c \mid a, b) &= \frac{\mathrm{p}(a, b, c)}{\mathrm{p}(a, b)} \\ &= \frac{\mathrm{p}(a)\,\mathrm{p}(b \mid a)\,\mathrm{p}(c \mid b)}{\mathrm{p}(a)\,\mathrm{p}(b \mid a)} \\ &= \mathrm{p}(c \mid b) \end{aligned} \tag{2.27}$$

If b is in the evidence, therefore, a linear path is inactive. The evidence blocks the path between a and c. This is intuitive in the sense that if we consider it as a causes b and b causes c", knowledge of b blocks the effect of a on c.

Diverging

For the diverging case, a and c have a common 'cause' c. a and c are not necessarily independent, so without b observed the configuration is active. With the joint defined as

$$\mathrm{p}(a, b, c) = \mathrm{p}(b)\,\mathrm{p}(a \mid b)\,\mathrm{p}(c \mid b) \tag{2.28}$$

we can show that a and c are independent given b

$$\begin{aligned} \mathrm{p}(c \mid a, b) &= \frac{\mathrm{p}(a, b, c)}{\mathrm{p}(a, b)} \\ &= \frac{\mathrm{p}(b)\,\mathrm{p}(a \mid b)\,\mathrm{p}(c \mid b)}{\mathrm{p}(b)\,\mathrm{p}(a \mid b)} \\ &= \mathrm{p}(c \mid b) \end{aligned} \tag{2.29}$$

Meaning that is b is part of the evidence the diverging configuration is inactive.

Converging

So far for the linear and diverging case the networks were both inactive when b is observed. The converging case is the opposite. The joint is

$$\mathrm{p}(a, b, c) = \mathrm{p}(a)\,\mathrm{p}(c)\,\mathrm{p}(b \mid a, c) \tag{2.30}$$

When b is not observed a and c are independent:

$$\begin{aligned} \mathrm{p}(a,b,c) &= \mathrm{p}(b \mid a,c)\,\mathrm{p}(a,c) \\ &= \mathrm{p}(a)\,\mathrm{p}(c \mid b)\,\mathrm{p}(b \mid a,c) \\ \mathrm{p}(a)\,\mathrm{p}(c)\,\mathrm{p}(b \mid a,c) &= \mathrm{p}(a)\,\mathrm{p}(c \mid b)\,\mathrm{p}(b \mid a,c) \\ \mathrm{p}(c) &= \mathrm{p}(c \mid b) \end{aligned} \tag{2.31}$$

And when b is observed a and c are not necessarily independent, making b unobserved the inactive case.

2.7.3 Plate Notation

Plate notation helps to indicate sampling in a directed graph (Buntine, 1994). Plates indicate repetition of the nodes inside the plate with the number of repeated nodes shown in the lower corner. For the coin-toss example, if we extend the model to any N coin tosses it would be unwieldy to show that in the style of figure 2.2, but using plate notation we more efficiently show the same information. Plates can be nested.

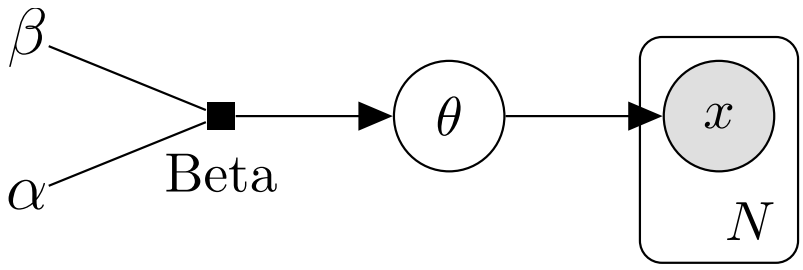


Figure 2.6: Bayes network with plate notation for determining the bias (θ) of a coin after N flips. The plate is a compact and convenient notation for representing repetition. The repetition is simply a representation of sampling; in this case our model assumes that we draw N samples from θ.

2.8 Some Useful Distributions

The exponential family of distributions is an important and flexible family of distributions that is often used in Bayesian modelling (Bishop, 2006). Notable distributions of the family include the normal, beta, Poisson, Bernoulli, multinomial and Dirichlet distributions, amongst others. The exponential family

provides an expressive set of densities that are broadly applicable and have desirable computational properties.

2.8.1 The Exponential Family

All the distributions in the family have distribution functions in the form over $\boldsymbol{x}$ and parametrised by $\boldsymbol{\eta}$

$$\mathrm{p}(\boldsymbol{x} \mid \boldsymbol{\eta}) = h(\boldsymbol{x})\exp\left\{\boldsymbol{\eta}^T T(\boldsymbol{x}) - a(\boldsymbol{\eta})\right\}. \tag{2.32}$$

Equation 2.32 is the *canonical* form for the exponential family. All of the distributions in the exponential family can be written in the canonical form. $\boldsymbol{\eta}$ are known as the *natural parameters* of the distribution; the function $a(\boldsymbol{\eta})$ acts to ensure that the distribution is normalised. $h(\boldsymbol{x})$ is known as the base measure and is a non-negative function that only depends on $\boldsymbol{x}$.

The exponential family has many desirable properties for Bayesian modelling. All the distributions have a conjugate priors in the exponential family and they have bounded sufficient statistics.

2.8.2 Sufficient Statistics

In the canonical form for the exponential family the sufficient statistic is given by the function $\mathrm{T}(\boldsymbol{x})$. The data contained in the sufficient statistic are all the data required to calculate the statistics (moments) of the distribution, via maximum likelihood or Bayesian methods (Bishop, 2006). An important property of the exponential family is that the dimensions of their sufficient statistics are bounded and do not grow with the amount of data observed. This is important for providing a compact representation for the real-world use of these distributions and ensures that they are computationally feasible for a wide range of models (Hogg *&* Craig, 1978).

2.8.3 The Normal Distribution

The normal or Gaussian is a well-known member of the exponential family. A continuous distribution for a random variable x, it is parametrised by a mean,

μ, and a standard deviation, σ (Peebles & Shi, 2001).

$$\mathcal{N}(x; \mu, \sigma^2) = \mathrm{p}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.33}$$

The density has a distinctive bell shape. The mean is a location parameter around which the density is centred and the standard deviation controls the spread of mass around the mean. Some example normals are shown in Figure 2.7, which illustrates how changing the mean and standard deviation changes the location and spread of the PDF.

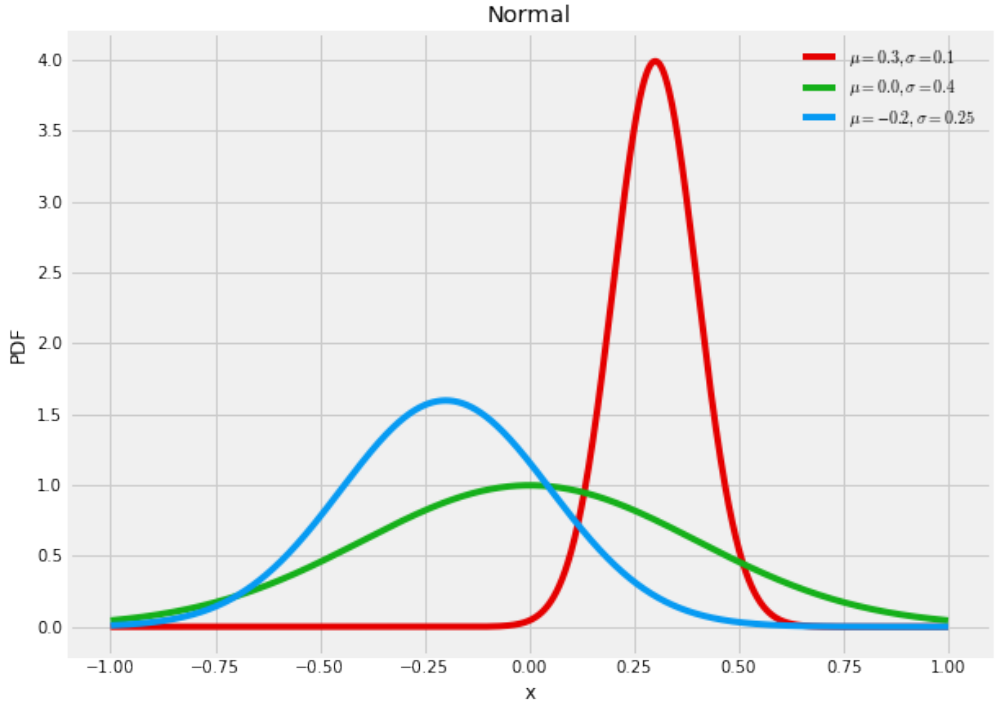


Figure 2.7: Probability density functions for three normal distributions with differing means and standard deviations. Note how the mean acts to locate the distribution and the standard deviation controls the spread around the mean.

2.8.4 The Binomial Distribution

The binomial distribution is a discrete distribution of the number of successes when carrying out n trials that have a success probability of μ. The binomial

distribution has the following probabilistic mass function:

$$\mathrm{p}(k \mid n, \mu) = \binom{n}{k} \mu^k (1-\mu)^{n-k} \tag{2.34}$$

where:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{2.35}$$

The binomial can be viewed as giving the probability of observing k heads when flipping a biased coin n times, where the bias μ is the probability of seeing heads. The PMF of various binomials with different success probabilities is shown in Figure 2.8.

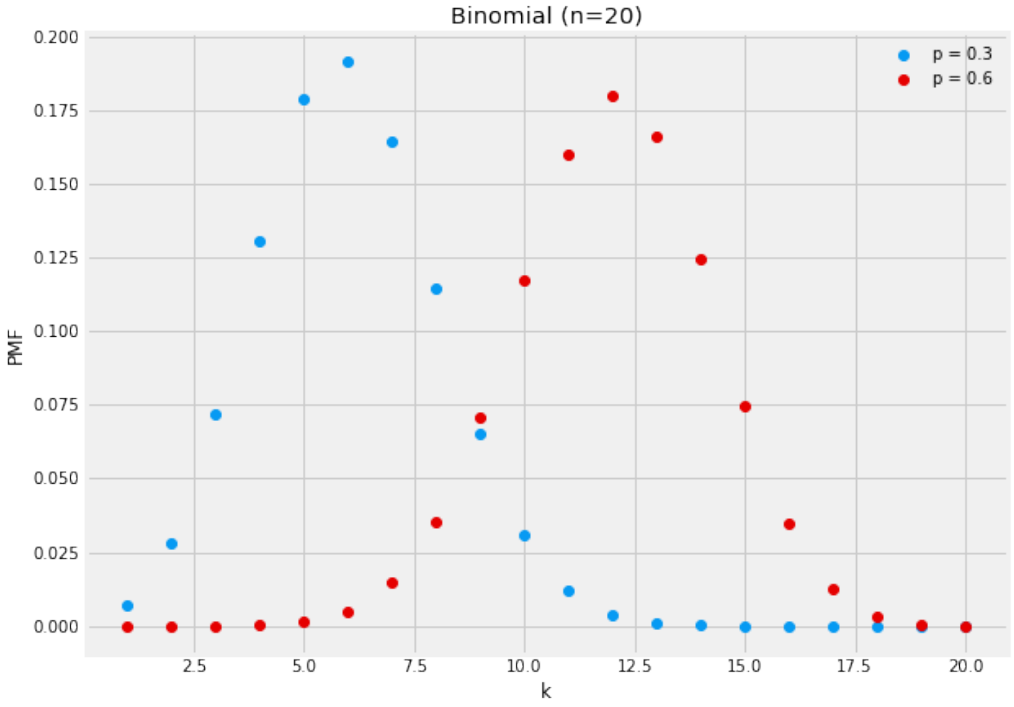


Figure 2.8: Probability mass functions for binomials with different success probabilities and twenty trials ($n = 20$). The binomial is discrete and thus has a PMF.

2.8.5 The Beta Distribution

The beta distribution is a continuous distribution over the range $[0, 1]$ and is the conjugate prior to the binomial distribution. It has two parameters, α and

β, and the PDF is given as:

$$\text{Beta}(x; \alpha, \beta) = \text{p}(x \mid \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}. \tag{2.36}$$

Where $B(\alpha, \beta)$ is a normaliser function that ensures that the distribution integrates to 1. The expected value is given by:

$$E[x] = \frac{\alpha}{\alpha + \beta}. \tag{2.37}$$

A sample from a beta distribution is a valid probability, making the beta distribution useful for modelling the likelihood of a binary event, such as success probability or the chance that a coin will come up heads.

2.8.6 The Multinomial Distribution

The multinomial distribution is a discrete distribution over K categories. It is parametrised by a vector $\boldsymbol{p}$ of probabilities where $\sum_{k=1}^{K} p_k = 1$ and p_k is the probability of getting category k, and a scalar n which is the number of categories returned from the multinomial. The PMF is:

$$\text{p}(\boldsymbol{x} \mid n, \boldsymbol{p}) = \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}. \tag{2.38}$$

2.8.7 The Categorical Distribution

The Categorical distribution is simply the multinomial for a single sample. In topic modelling and information retrieval, the Multinomial and Categorical distributions are often used interchangeably (Blei & Lafferty, 2009).

2.8.8 The Dirichlet

The Dirichlet distribution is the multivariate extension to the Beta distribution. The Dirichlet is parametrised by a K-dimensional vector $\boldsymbol{\alpha}$.

$$\text{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}) = \text{p}(\boldsymbol{x} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} x_k^{\alpha_k - 1} \tag{2.39}$$

Where the normaliser is given as:

$$B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)}. \tag{2.40}$$

The Dirichlet is a (conjugate) prior to the multinomial distribution. Samples from the Dirichlet are valid probability vectors

$$\begin{aligned} \boldsymbol{x} &\sim \mathrm{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}) \\ \sum_{k=1}^{K} x_k &= 1 \end{aligned} \tag{2.41}$$

This constraint causes the Dirichlet to be constrained to a $K-1$ probability *simplex*, shown as the blue triangle in figure 2.9. A simplex is the generalisation of a tetrahedral region or triangle to k dimensions (Boyd & Vandenberghe, 2004). A 1-dimensional simplex is a line segment, a 2-dimensional simplex is a triangle, and a 3-dimensional simplex is a tetrahedron. The probability simplex is the set of all vectors with non-negative elements that sum to one. It is common to visualise the Dirichlet as a 3 dimensional Dirichlet that then lies on the 2-simplex. Figures 2.9, 2.10, and 2.11 show the PDF on the simplex.

The Dirichlet parameter $\boldsymbol{\alpha}$ can be given as the element-wise product of a mean $\boldsymbol{m}$ and scalar precision parameter, so that

$$\alpha_k = m_k \cdot s, \quad \forall K. \tag{2.42}$$

The mean $\boldsymbol{m}$ is a localizing parameter of the Dirichlet and is the point on the simplex around which the Dirichlet is centred and is calculated by normalizing $\boldsymbol{\alpha}$ so that the mean sums to 1:

$$m_k = \frac{\alpha_k}{\sum_{k'=1}^{K} \alpha_k}, \quad \forall K. \tag{2.43}$$

The concentration parameter or precision s controls the sparsity of the Dirichlet:

$$s = \sum_{k=1}^{K} \alpha_k. \tag{2.44}$$

There are three noteworthy states for the Dirichlet distribution. When the concentration parameter is equal to the dimension of the Dirichlet, $s = K$, the PDF is uniform over the $K-1$ simplex.

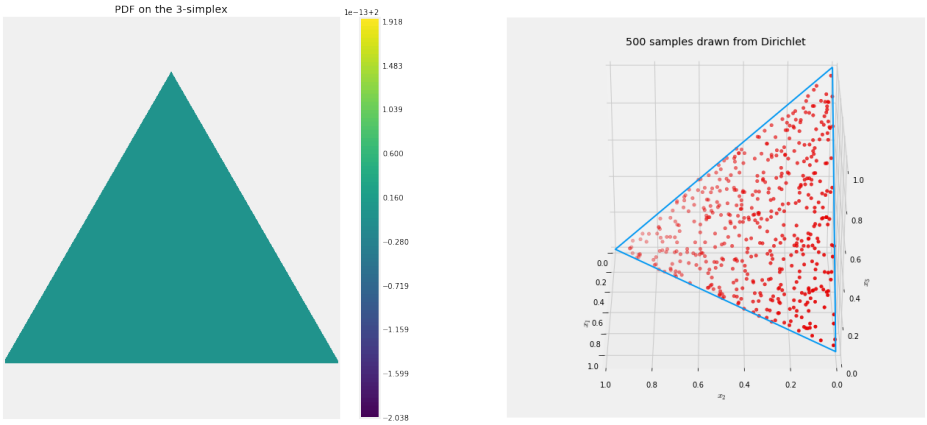


Figure 2.9: A uniform Dirichlet. The PDF on the 3-simplex is shown on the left and 500 samples from the Dirichlet on the right. In this case the dimension of the Dirichlet is three ($K = 3$); the PDF lies on a 2-simplex which is a triangle on a plane. The PDF is even over this 2-simplex.

When the concentration parameter is greater than K ($s > K$), the PDF becomes concentrated around the mean.

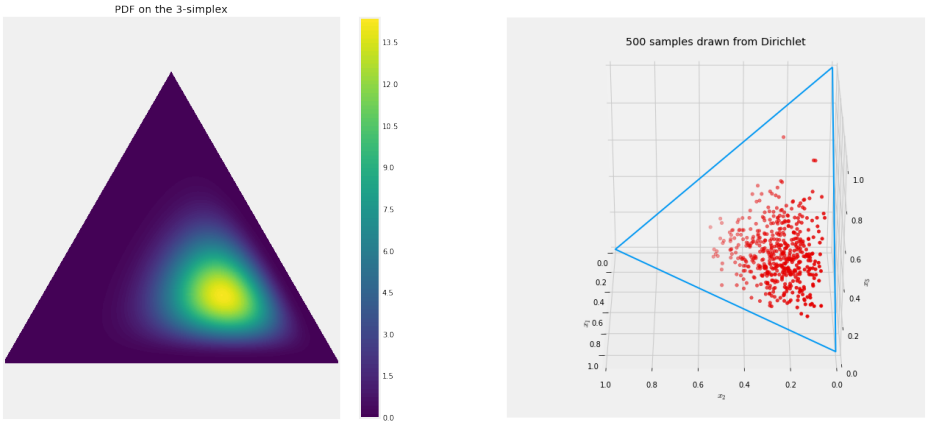


Figure 2.10: When the concentration parameter is above the dimension K (in this case 3), the Dirichlet concentrates its mass around the mean.

When the concentration parameter is less that the dimension $s < K$, the Dirichlet PDF pushes away from the mean into the 'corners' of the $K - 1$

simplex.

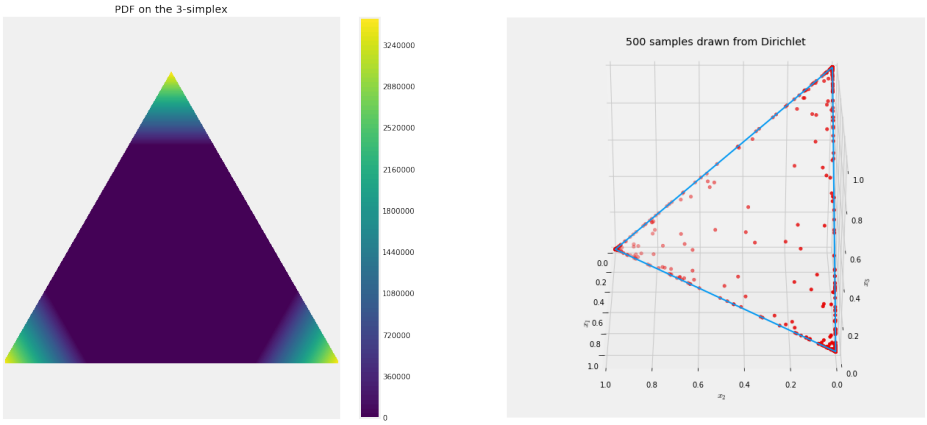


Figure 2.11: When the concentration parameter is below K, the PDF mass moves into the corners of the $K-1$ simplex.

Sampling from the Dirichlet

We wish to sample a random vector $\boldsymbol{x}$ from a Dirichlet with parameter $\boldsymbol{\alpha}$. If we have a source of Gamma distributed random variables, we can sample $\boldsymbol{x}$ by first drawing K independent random samples from gamma distributions with the density:

$$y_i \sim \text{Gamma}(\alpha_i, 1) = \frac{y_i^{\alpha_i - 1} e^{-y_i}}{\Gamma(\alpha_i)} \tag{2.45}$$

and then normalise to get $\boldsymbol{x}$

$$x_k = \frac{y_k}{\sum_{k=1}^{K} y_k}. \tag{2.46}$$

Estimating Parameters

Minka (2000) shows how to estimate the parameters of the Dirichlet. The relevant methods are summarised here. In general, given some observed data $\mathcal{D} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, the Dirichlet has a bounded sufficient statistic

$$\log(\bar{x}_k) = \frac{1}{N} \sum_{n=1}^{N} x_{nk}. \tag{2.47}$$

The log-likelihood of the Dirichlet is for the observed data $\mathcal{D}$ is

$$\log \mathrm{p}(\mathcal{D} \mid \boldsymbol{\alpha}) = N \log \Gamma \left(\sum_{k=1}^{K} \alpha_k \right) - N \sum_{k=1}^{K} \log \Gamma \left(\alpha_k \right) + N \sum_{k=1}^{K} \left(\alpha_k - 1 \right) \log \bar{x}_k. \tag{2.48}$$

Linear Time Newton-Raphson Algorithm for Estimating Parameters

The Newton-Raphson optimization technique finds a stationary point of a function with a gradient g and Hessian $\boldsymbol{H}$ by iterating:

$$\alpha_{new} = \alpha_{old} - H \left(\alpha_{old} \right)^{-1} g \left(\alpha_{old} \right) \tag{2.49}$$

The derivative (gradient) of of the log-likelihood with respect to α_k is

$$g_k = \frac{\partial \log p \left(\mathcal{D} \mid \boldsymbol{\alpha} \right)}{\partial \alpha_k} = N \psi \left(\sum_{k=1}^{K} \alpha_k \right) - N \psi \left(\alpha_k \right) + N \log \bar{x}_k \tag{2.50}$$

The second-derivatives (Hessian) of the log-likelihood with respect to $\boldsymbol{\alpha}$ are

$$\frac{\partial \log p \left(\mathcal{D} \mid \boldsymbol{\alpha} \right)}{\partial \alpha_k^2} = N \psi' \left(\sum_{k=1}^{K} \alpha_k \right) - N \psi' \left(\alpha_k \right) \tag{2.51}$$

$$\frac{\partial \log p \left(\mathcal{D} \mid \boldsymbol{\alpha} \right)}{\partial \alpha_k \partial \alpha_j} = N \psi' \left(\sum_{k=1}^{K} \alpha_k \right) \qquad (k \neq j) \tag{2.52}$$

The Hessian, $\boldsymbol{H}$, can be written in matrix form as

$$\boldsymbol{H} = \boldsymbol{Q} + \mathbf{1} z \mathbf{1}^T \tag{2.53}$$

where

$$z = N \psi' \left(\sum_{k=1}^{K} \alpha_k \right) \tag{2.54}$$

and $\boldsymbol{Q}$ has elements

$$q_{jk} = -N \psi' \left(\alpha_k \right) \delta \left(j - k \right) \tag{2.55}$$

For Newton-Raphson, the Hessian still needs to be inverted. In general this matrix inversion causes the algorithm to scale to $O(N^3)$, but in this case

the Hessian has a special structure that allows for inversion in linear time. Applying the matrix inversion lemma (appendix **??**) to $\boldsymbol{H}$, we obtain

$$\boldsymbol{H}^{-1} = \boldsymbol{Q}^{-1} - \frac{\boldsymbol{Q}^{-1}\mathbf{1}\mathbf{1}^T\boldsymbol{Q}^{-1}}{z^{-1} + \mathbf{1}^T\boldsymbol{Q}^{-1}\mathbf{1}} \tag{2.56}$$

We then multiply by the gradient to get

$$\left(\boldsymbol{H}^{-1}\boldsymbol{g}\right)_k = \frac{g_k - b}{q_{kk}} \tag{2.57}$$

where

$$b = \frac{\mathbf{1}^T\boldsymbol{Q}^{-1}\boldsymbol{g}}{1/z + \mathbf{1}^T\boldsymbol{Q}^{-1}\mathbf{1}} = \frac{\sum_j g_j/q_{jj}}{1/z + \sum_j + \sum_j 1/q_{jj}} \tag{2.58}$$

Therefore one Newton step for α_k is

$$\alpha_k^{new} = \alpha_k^{old} - \frac{g_k - b}{q_{kk}} \tag{2.59}$$

Estimating the Mean

If we fix the precision, s, and get the likelihood for $\boldsymbol{m}$ alone

$$\mathrm{p}(D \mid \boldsymbol{m}) \propto \left(\prod_{k=1}^{K} \frac{\exp(sm_k \log \bar{x}_k)}{\Gamma(sm_k}\right)^N \tag{2.60}$$

Parametrise with the unconstrained vector $\boldsymbol{z}$,

$$m_k = \frac{z_k}{\sum_{j=1}^K z_j} \tag{2.61}$$

to get the gradient:

$$\frac{\partial\, \mathrm{p}(D \mid \boldsymbol{m})}{\partial z_k} = \frac{Ns}{\sum_{j=1}^K z_j}\left(\log \bar{x}_k - \sum_{j=1}^{K} m_j^{old}(\log \bar{x}_j) - \psi\left(sm_j^{old}\right)\right) \tag{2.62}$$

The MLE can be computed by the fixed point iteration

$$\alpha_k = \psi^{-1}\left(\log \bar{p}_k - \sum_{j=1}^{K} m_j^{old}\left(\log \bar{x}_j - \psi\left(sm_j^{old}\right)\right)\right) \tag{2.63}$$

and

$$m_k^{new} = \frac{\alpha_k}{\sum_{j=1}^K \alpha_j} \tag{2.64}$$

2.9 Summary

This chapter provided a brief review of certain Bayesian concepts. Some important concepts are how the Bayes network not only shows the dependency structure but allows for a convenient factorisation of the posterior. This factorisation is used by the approximate inference algorithms in Chapter 5. The exponential family is also very important when discussing inference, as it is common to restrict distributions into belonging to the exponential family. Another important concept is how the prior acts to regularise the model and prevent overfitting. We saw in the example in Section 2.4 how the data needs to overwhelm the prior with sufficient evidence.

Chapter 3

Recommender Systems

3.1 Collaborative Filtering and Recommendation Tasks

3.1.1 Recommender System Problem Definition

The recommender system problem consists of a set of users $\boldsymbol{U}$ and a set of items $\boldsymbol{I}$. The goal is one of information retrieval, where the recommender is tasked with finding undiscovered items that provide maximum utility to the user. Commonly utility is considered as a measure of enjoyment. This is intuitive; the recommender is tasked with discovering enjoyable items for a user, be it media, web pages or e-commerce products. Another related utility is engagement, where the goal is to retrieve items for a user that the user would select had they sought it out on their own. This highlights an important aspect of recommender systems, namely their role in filtering, where they assist users in finding relevant items in the often vast selections common in the web age.

Users provide feedback on items and the observed feedback user u gave to item i is denoted as $r_{u,i}$. For a system consisting of U users and I items, we

can construct $\boldsymbol{R}$ as a $U \times I$ feedback matrix.

$$\boldsymbol{R}_{U\times I} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,I} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,I} \\ \vdots & \vdots & \ddots & \vdots \\ r_{U,1} & r_{U,2} & \cdots & r_{U,I} \end{pmatrix} \tag{3.1}$$

Most of the possible $u \times i$ pairs are unobserved (denoted by $r_{u,i} = \emptyset$) so the feedback matrix will typically be sparse. $\boldsymbol{I}_u$ is the set of items for which user u has given feedback and $\boldsymbol{U}_i$ is likewise the set of users that have given feedback for item i.

While not normally stated, the feedback matrix representation assumes that the feedback data is exchangeable (section 2.6). We only take note that we observed the feedback; no sequence or timing information is preserved in the matrix.

		Items ($\boldsymbol{I}$)							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Users ($\boldsymbol{U}$)	**1**	3	$\emptyset$	2	5	$\emptyset$	4	$\emptyset$	1
	2	$\emptyset$	1	4	4	$\emptyset$	$\emptyset$	5	$\emptyset$
	3	1	2	$\emptyset$	$\emptyset$	3	$\emptyset$	$\emptyset$	2
	4	2	3	$\emptyset$	5	2	$\emptyset$	4	2
	5	3	$\emptyset$	3	$\emptyset$	3	4	$\emptyset$	1

Figure 3.1: A simple example of a feedback matrix

3.1.2 Types of Feedback

The feedback matrix seen in figure 3.1 is an example of *explicit* feedback. This is when users directly indicate their preference for items, e.g. ratings. The most basic form of explicit feedback is a binary variable indicating like-dislike. A more common type is a ratings scale feedback where users indicate their preference for items by giving a rating on some scale, or range. For explicit feedback a user is represented as a vector of observed ratings $\boldsymbol{r}_u$, which is simply a row from the feedback matrix.

The second type of feedback is found by observing user behaviour. This *implicit* feedback consists of observations such as *user clicked a link* or *user purchased an item*. Note that all explicit feedback has associated implicit feedback, and each rating also carries the observation that the user rated the item. With the growth of the web, practical recommender systems often have access to large amounts of implicit data. When working with implicit data a user can be represented as a co-occurrence vector $\boldsymbol{y}_u$ where $y_{u,i}$ is a count of interactions between user u and item i.

3.1.3 Recommendation Tasks and Evaluation

We will focus on a class of recommender techniques commonly called *collaborative filtering*. Collaborative-filtering recommender systems recommend items to users based on the behaviour and history of other users (Schafer *et al.*, 2007). The general task of the collaborative-filtering recommender system is, given the feedback matrix, to present a filtered list of items to the user that maximises their utility. Companies such as Amazon (Smith & Linden, 2017), NetFlix (Gomez-Uribe & Hunt, 2015), YouTube (Davidson *et al.*, 2010) and many others can evaluate their recommender systems online by comparing real users' experiences with new recommenders. However in the absence of an active user base and to develop new algorithms and approaches offline, an evaluation of recommender systems is also required. There are many possible ways of evaluating recommender systems using static datasets and the methods have evolved with the field. Early evaluations were focused on accuracy metrics that made use of explicit data. This was in part driven by the fact that in many cases explicit data was more readily available. One such example occurred in the early days of NetFlix, before streaming, where the only feedback received was ratings that users gave DVDs they had rented (Gomez-Uribe & Hunt, 2015).

With the explosive growth of the web and increases in computing power, more and more implicit data became available. This, coupled with criticisms of accuracy metrics (McNee *et al.*, 2006), led to the field exploring other ways of evaluating recommender systems such as evaluation via ranking (Vargas &

Castells, 2011), and techniques from information retrieval.

Evaluating recommender systems offline is a difficult task for many reasons. Many recommender systems are domain specific or designed for specific datasets, and may struggle on datasets from different fields, or datasets that have different properties. One such property may be the ratio of users to items; some algorithms are designed specifically for datasets with more users than items and may perform poorly where the opposite is true (Herlocker *et al.*, 2004). A common dataset for the evaluation of recommender systems was the NetFlix Prize dataset (Bennett *et al.*, 2007). Unfortunately, for privacy reasons it is no longer available. A similar, high quality and commonly used dataset is the MovieLens movie ratings dataset (Harper & Konstan, 2015).

Rating Prediction

One of the earliest and still common prediction tasks is *rating prediction*. The goal for rating prediction is to predict the rating a given user would give an item. The system can then predict all the ratings on the unconsumed items for a user and generate a list of items by sorting all items by their predicted rating. The assumption behind rating prediction is that the ratings supplied by a user accurately reflect their preference. By better predicting user preference, the system can discover items that the user will enjoy. Most recommenders present a sorted list of items that it predicts are relevant, but they do not typically present predicted ratings alongside items. This makes rating prediction somewhat of an interesting case as it is not scoring the system on the exact task it is performing, but on a proxy task.

For a predicted rating $\hat{r}_i$ on item i, the accuracy of the prediction is commonly measured via *Root-Mean-Square-Error* (RMSE), which is defined for the items in a test set I':

$$\text{RMSE} = \sqrt{\frac{1}{I'}\sum_{i'=1}^{I'}(r_{i'} - \hat{r}_{i'})^2}. \tag{3.2}$$

RMSE is a measure of how close the predicted ratings were to the true ratings in the test set. The squaring of the error in RMSE causes it to punish

large errors more than small errors. A drawback of RMSE based evaluation is that it does not resemble how recommenders are commonly used in practice. The focus on rating prediction by the field of recommender systems has been criticised, and the assumption that the accuracy of the rating prediction improved recommendation quality may be misleading (McNee *et al.*, 2006). An important piece of feedback from the commercial side of recommender systems was NetFlix's own feedback after the NetFlix Prize competition. NetFlix shifted away from focusing on rating prediction and toward incorporating more implicit data (Gomez-Uribe & Hunt, 2015).

Item prediction

An alternative recommendation method to rating prediction is to task the recommender with generating the recommended list of items directly; this is known as *item prediction*. By taking the top N recommended items we can evaluate them based on how relevant they are to the selected user. This is an information retrieval problem and we can use metrics from that field. This type of evaluation is commonly called Top-N recommendation due to its evaluation of the recommender's top N most relevant items. To evaluate top-N recommendation, we use two measures: *precision* and *recall* from information retrieval (Ting, 2011). Precision and recall are defined as follows:

$$precision = \frac{Total\ number\ of\ relevant\ items\ retrieved}{Total\ number\ of\ items\ retrieved} \tag{3.3}$$

$$recall = \frac{Total\ number\ of\ relevant\ items\ retrieved}{Total\ number\ of\ relevant\ items} \tag{3.4}$$

We can use a confusion matrix with two classes where relevant items are in the positive class and irrelevant items in the negative, giving the confusion matrix as:

	actual positive	actual negative
predicted positive	TP	FP
predicted negative	FN	TN

Figure 3.2: Confusion Matrix

In the context of recommendation precision and recall can then be defined as (Davis & Goadrich, 2006):

$$precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$recall = \frac{TP}{TP + FN}. \tag{3.6}$$

To evaluate both at once we can use the *F1* score, which is a linear blend of recall and precision

$$\begin{aligned} \text{F1} &= \frac{1}{\frac{1}{recall} + \frac{1}{precision}} \\ &= 2\frac{recall \times precision}{recall + precision} \end{aligned} \tag{3.7}$$

A problem with recommendation is that we have a static snapshot of user behaviour and do not know the set of all relevant items for a user. Hypothetically a recommender could evaluate poorly offline if it finds relevant items for a user that have never been discovered by the user.

3.2 Factors for Effective Recommendation

There are many other factors that affect recommendation that are not covered by accuracy metrics. Many of these other factors are difficult or impossible to measure but need to be considered when evaluating a recommender model or algorithm.

Recommender systems are an interesting class of system because the model is only a small part of a larger system involving user interaction. Many other factors influence an effective recommender apart from accuracy metrics. Some of these factors are related to performance: an algorithm or model may need to scale to millions of users and items. Scaling in representation is not the only performance characteristic; recommenders are live systems with users continually adding feedback and new users joining. Models that are slow to update will have stale recommendations and take extra time to start recommending to new users.

An important concept for recommendations is if the recommendation can be accompanied by an explanation of why the system recommended what it did. This explanation is an important part of building trust with users (Ramezani *et al.*, 2008), and users have been shown to prefer recommendations that are explained (Gomez-Uribe & Hunt, 2015). The explanation can be as simple as 'users similar to you enjoyed these items' or 'items similar to this item you recently chose'.

Related to scalability is how the system handles the often extreme sparsity of the input data (Adomavicius & Tuzhilin, 2005). Sparsity introduces challenges in representation, where most of the possible values in the feedback matrix are empty and representing them becomes computationally intractable. This also creates an issue of coverage, where so little is observed of the possible user-item pairs and most of the observations are concentrated on relatively few items. This concentration causes the system to begin ignoring large parts of the item corpus (Konstan *et al.*, 1998). The system then essentially starts finding better combinations of popular items instead of personalising to the user. The recommendations start to lack diversity as every user gets recommended the same popular items and cannot discover new items. It is then very important for the recommender to cover less popular items and personalise recommendations to the user.

Sparsity also exposes the problem of recommending to users with very little or no history; this is known as the *cold start* problem (Schein *et al.*, 2002). Collaborative filtering relies on user histories to make recommendations, but if a user has just joined the system or has a very small history, recommendation becomes challenging or even impossible for some systems. Some systems can natively handle new users, but others require special rules for users without a history.

3.3 Neighbourhood Methods

The first collaborative filtering recommendation systems were based on heuristic methods that do not specify a model. Instead, recommendations are gener-

ated using all the observations in the database. For this reason these methods are commonly known as memory-based methods. The intuition behind these methods is simple: when recommending items to a user, find similar users and recommend items based on items the similar users have rated highly. This approach is simple and intuitive and it achieves reasonably good results for forced prediction, which has resulted in it remaining popular even against more recent approaches. Neighbourhood methods generate recommendations via rating prediction.

The simplest neighbourhood technique generates a predicted rating $\hat{r}_{u,i}$ for user u and item i by first selecting a neighbourhood of users $\boldsymbol{V}$ and then predicting the rating as the mean rating given by the users in the neighbourhood.

$$\hat{r}_{u,i} = \frac{1}{|V|} \sum_{v=1}^{V} r_{v,i} \tag{3.8}$$

The neighbourhood $\boldsymbol{V}$ can be as simple as consisting of users that have rated item i, in which case the neighbourhood method is simply recommending based on item means. A better approach is to use a similarity measure between users to build the neighbourhood.

The similarity measure is a function denoted $\text{sim}(\boldsymbol{r}_u, \boldsymbol{r}_v)$ that, given two user rating vectors return a value in the range $[-1, 1]$ where -1 indicates the users have exactly opposite histories and $+1$, indicates their histories are exactly the same.

We can use the similarity measure to predict the rating as a weighted average for the users in $\boldsymbol{V}$, where the item ratings are weighted based on the similarity between the selected user and the user from the neighbourhood. This way the ratings of more similar users contribute more than dissimilar users.

$$r_{u,i} = \frac{1}{\sum_{v=1}^{V} \text{sim}(\boldsymbol{r}_u, \boldsymbol{r}_v)} \sum_{v=1}^{V} \text{sim}(\boldsymbol{r}_u, \boldsymbol{r}_v) r_{v,i} \tag{3.9}$$

The neighbourhood can also be filtered down further from all the users that have rated item i by only keeping the K most similar users to the selected user

u in $\boldsymbol{U'}$; these top-K neighbourhoods let one control the size of the neighbourhood that is used.

3.3.1 Normalisation

An obstacle to this approach is that not all users use the rating scale in the same way. To remedy this we can normalise the ratings to keep the preference to rating mapping consistent between users (Ekstrand *et al.*, 2011). For a user represented as a rating vector $\boldsymbol{r}_u$ we can normalise around the user mean μ_u by subtracting the mean from each rating so that the user feedback represented as deviation from the mean:

$$r_{u,i}^{new} = r_{u,i} - \mu_u. \tag{3.10}$$

To further normalise the ratings to account for the differences in how users spread their ratings over the rating scale, the user's ratings can be centred around the user mean and then scaled relative to the user's standard deviation σ_u

$$r_{u,i}^{new} = (r_{u,i} - \mu_u)/\sigma_u. \tag{3.11}$$

When predicting ratings the predicted ratings must be scaled back to the original rating scale.

3.3.2 Similarity Measures

The two most common and effective similarity measures are cosine similarity and Pearson correlation.

Cosine similarity is a vector similarity measure that measures the cosine of the angle between two vectors.

$$\text{sim}(\boldsymbol{r}_u, \boldsymbol{r}_v) = cos(\theta) = \frac{\sum_{i=1}^{I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i=1}^{I} r_{u,i}^2}\sqrt{\sum_{i=1}^{I} r_{v,i}^2}} \tag{3.12}$$

Pearson correlation coefficient is a linear correlation measure where $\boldsymbol{I'} = \boldsymbol{I}_u \cap \boldsymbol{I}_v$

$$\text{sim}(\boldsymbol{r}_u, \boldsymbol{r}_v) = \rho_{\boldsymbol{r}_u, \boldsymbol{r}_v} = \sum_{\forall(\mathbf{I'})} \frac{(r_{u,i} - \mu_u)(r_{v,i} - \mu_v)}{\sqrt{(r_{u,i} - \mu_u)^2}\sqrt{(r_{v,i} - \mu_v)^2}}. \tag{3.13}$$

μ_u is the feedback average for user u over $\boldsymbol{I}_u \cap \boldsymbol{I}_v$.

Both of these similarity measures give jump to the edges of their range values when the user histories are small. This causes users with small histories to seem very similar to other users even though they only have a small overlap in histories. This effect creates issues in cold start scenarios. This causes neighbourhood models to be generally unsuitable for cold start scenarios and there needs to be specific strategies for new users, or a hybrid system is required (Lam *et al.*, 2008).

3.3.3 Usage and Drawbacks

Neighbourhood models are conceptually simple and intuitive, and are commonly recommended as a starting point for recommender systems. However, they are less accurate than matrix factorisation for rating prediction (Koren *et al.*, 2009) and perform very poorly for top-N item prediction (Cremonesi *et al.*, 2010).

The recommendations generated are explainable as 'users similar to you enjoyed:', but because no explicit model is learned, the system cannot gain any insight into user behaviour or the relationship between items. Neighbourhood models are also difficult to scale and much of the desired simplicity of the approach is lost with complex extensions to enable scaling. Despite these drawbacks there are approaches that successfully employ neighbourhood techniques. *Amazon.com* for example used what they called item-to-item collaborative filtering (Linden *et al.*, 2003). This combines many approaches including the traditional collaborating filtering approach into a hybrid, scalable recommender.

3.4 Model-based Recommender Systems

Matrix factorisation models gained popularity over neighbourhood models over the course of the *Netflix Prize* due to the fact that they typically perform better than neighbourhood models for rating prediction (Koren *et al.*, 2009).

3.5 Matrix Factorization

Matrix factorisation models are simple linear latent factor models that seek to map users and items to a common latent feature space. The core idea is that by restricting the dimensionality of the feature space, the model is forced to generalise the users and items when mapping to the space. Users and items start in the high dimensional and sparse feedback space (vectors from the feedback matrix); the model then maps them to a low-dimensional feature space. The feature space ideally maps similar users and similar items closely together.

The core matrix factorisation model maps users and items to a joint feature space with dimension K (where K is much smaller than U or I). The mapping is performed in such a way that in the latent feature, space user-item interactions are modelled as dot products. In this space a user u is represented by a K dimensional feature vector $\boldsymbol{v}_u$. Each item i is likewise represented as the vector $\boldsymbol{y}_i$. A user-item interaction is modelled as the dot product of these two vectors:

$$r_{u,i} = \boldsymbol{v}_u^T \boldsymbol{y}_i \tag{3.14}$$

When the user-item pair is unobserved, this dot-product gives the model's predicted feedback for the pair. We can collect all the user vectors into a $U \times K$ user-feature matrix $\boldsymbol{V}$, and likewise collect the item vectors into a $I \times K$ item-feature matrix $\boldsymbol{Y}$. The model's predicted feedback $\hat{\boldsymbol{R}}$ for all the items in the feedback matrix can be given as (shown visually in figure 3.3:

$$\hat{\boldsymbol{R}} = \boldsymbol{V}\boldsymbol{Y}^T \tag{3.15}$$

This illustrates why the models are referred to as matrix factorisation models, as the feedback matrix has been factorised into two low-rank feature matrices. In fact, one of the most popular ways of learning the model is via low-rank approximations via matrix factorisation techniques such as Singular Value Decomposition (SVD).

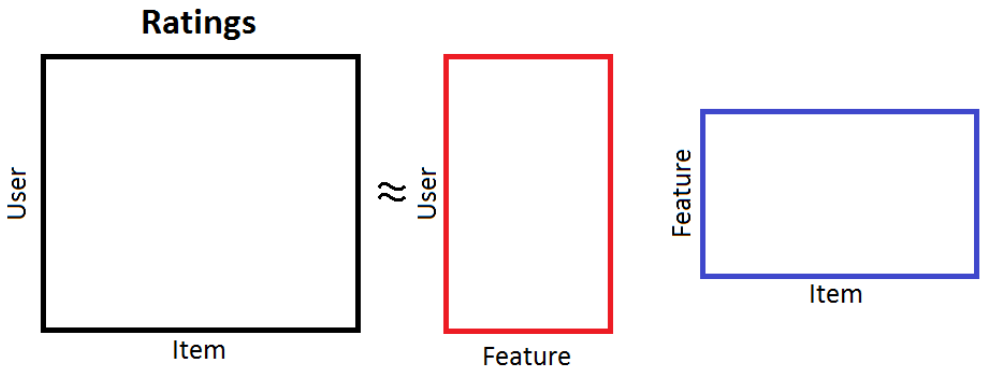


Figure 3.3: Matrix factorisation based recommendation.

3.5.1 Singular Value Decomposition

An early and simple approach to matrix factorisation recommender systems is to use *singular value decomposition* (SVD) to decompose the feedback matrix. This approach is sometimes called *PureSVD* (Cremonesi *et al.*, 2010) to differentiate from later techniques that built on the SVD foundation. After decomposing the feedback matrix using SVD, PureSVD constructs a low-rank approximation to the original feedback matrix via user- and item-feature matrices that capture as much variance in the dataset as possible.

For a $U \times I$ real matrix $\boldsymbol{R}$, the singular value decomposition defines a factorisation of the matrix into three matrices of the form (Golub & Reinsch, 1970):

$$\boldsymbol{R}_{U\times I} = \boldsymbol{A}_{U\times U}\boldsymbol{\Sigma}\boldsymbol{B}^T_{I\times I} \tag{3.16}$$

where $\boldsymbol{A}$ is a $U \times U$ real unitary ($\boldsymbol{A}$ has orthogonal columns so that $\boldsymbol{A}\boldsymbol{A}^T = \mathbf{I}$) matrix and $\boldsymbol{B}$ is likewise a $I \times I$ real unitary matrix. The matrix $\boldsymbol{\Sigma}$ is a diagonal matrix with the singular values of $\boldsymbol{R}$ on the diagonal. The singular values are the positive square roots of the eigenvalues of $\boldsymbol{R}\boldsymbol{R}^T$ (Klema & Laub, 1980). By convention the singular values are sorted in descending order.

To obtain a rank-K approximation of $\boldsymbol{R}$, all but the K largest singular values in $\boldsymbol{\Sigma}$ are set to zero to get a new singular matrix $\hat{\boldsymbol{\Sigma}}$. By exploiting the fact that the zeroes on the diagonal of $\hat{\boldsymbol{\Sigma}}$ will set the corresponding columns in the

left matrix to all zero and the corresponding rows in the right; we can fold $\hat{\boldsymbol{\Sigma}}$ into $\boldsymbol{A}$ and $\boldsymbol{B}^T$ by dropping the zero columns in both matrices and multiplying $\hat{\boldsymbol{\Sigma}}$ into one of them, to get rank-K approximate feature matrices $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{Y}}^T$.

The rank-K SVD can now be rewritten to give a rank-K approximation of $\boldsymbol{R}$:

$$\hat{\boldsymbol{R}} = \hat{\boldsymbol{V}}\hat{\boldsymbol{Y}}^T \tag{3.17}$$

where $\hat{\boldsymbol{R}}$ is the $U \times I$ rank-K approximation of the original matrix $\boldsymbol{R}$, $\hat{\boldsymbol{V}}$ is a $U \times K$ matrix and $\hat{\boldsymbol{Y}}^T$ is a $I \times K$ matrix. $\hat{\boldsymbol{R}}$ is a rank-K approximation to $\boldsymbol{R}$, which minimizes the Frobenius norm of the difference between $\boldsymbol{R}$ and $\hat{\boldsymbol{R}}$ (Weisstein, 2002), which is the squared distance for each user u for all items I' that exist for that user (Weisstein, 2003)

$$||\boldsymbol{R} - \hat{\boldsymbol{R}}|| = \sqrt{\sum_{u=1}^{U}\sum_{i=1}^{I'}(r_{u,i} - \hat{r}_{u,i})^2}. \tag{3.18}$$

Equivalently we can say that the low-rank approximation finds the approximate form that captures as much of the original variance as possible. By discarding dimensions that cause a minimal change in distance between the approximation and original representation, we discard dimensions that do not explain much variance. The dimensions retained should in theory provide good general representations.

When applied to the feedback, the matrices $\hat{\boldsymbol{Y}}^T$ are $K \times I$ and can be thought of as feature matrices, where the SVD maps users and items into a K-dimensional feature space. If the approach finds a good feature space, similar users and items should be close together when mapped to the space.

Drawbacks

The standard SVD is only defined for dense matrices, so for recommendation tasks where the feedback matrix is typically extremely sparse the naive SVD approach does not apply. One method to address this is to fill in the missing values with placeholder values. The inputation is a major obstacle, as incorrect inputation can distort the data. Some simple approaches are to fill the

missing values with zeros or averages. The SVD approach then runs into the additional computational hurdle of representing the entire feedback matrix. The feedback matrix is $U \times I$ and for large numbers users and items begins to become computationally intractable. For example, Amazon.com had 29 million customers and several million items in 2003 (Linden *et al.*, 2003), which would lead to a feedback matrix with trillions of entries.

3.5.2 Approximate Versions

Although the *PureSVD* approach generally performs well on rating prediction (Koren *et al.*, 2009), it is computationally expensive. To address this drawback and to better handle sparse matrices, versions that learned the singular vectors directly from the observed values were developed. An original approach was given by Brandon Webb as the 'Simon Funk' approximate SVD (Bennett *et al.*, 2007), which uses stochastic gradient descent. The approach gained notoriety by jumping to third place on the leaderboard for the *Netflix Prize* despite its simplicity (Funk, 2006). Many recommender matrix factorisation approaches built on this foundation after the conclusion of the NetFlix Prize. Gradient based methods are not the only approximate method for learning the feature matrices, and alternating least squares can also be used (Koren *et al.*, 2009).

The central idea is to use gradient descent on the error between the predicted and observed ratings to learn the feature matrices directly (Funk, 2007) and constuct the matrix factorisation model. For the model the predicted feedback matrix is given as:

$$\hat{\boldsymbol{R}} = \boldsymbol{V}\boldsymbol{Y}^T \tag{3.19}$$

For an observed rating r and a predicted rating $\hat{r}$ the square error is then:

$$E^2 = (r - \hat{r})^2 \tag{3.20}$$

The predicted rating is the dot product of a user's feature vector $\boldsymbol{v}$ and an item's feature vector $\boldsymbol{y}$.

$$\hat{r} = \boldsymbol{v}^T\boldsymbol{y} = \sum_{k=1}^{K} v_k y_k \tag{3.21}$$

We wish to learn each of the K features in the features matrices. To do so we take the partial derivative of the error with respect to the k-th user feature v_k:

$$\begin{aligned} \frac{\partial}{\partial v_k} E^2 &= \frac{\partial}{\partial v_k}(r - v_k \cdot y_k)^2 \\ &= -2Ey_k \end{aligned} \tag{3.22}$$

The gradient descent update for $\boldsymbol{v}_k$ with learning rate λ is:

$$\begin{aligned} v_k^{new} &= v_k^{old} - \lambda \frac{\partial}{\partial v_k} E^2 \\ &= v_k^{old} + \lambda E y_k \end{aligned} \tag{3.23}$$

We can then do the same for the item feature $\boldsymbol{y}_k$, by first getting the partial derivative of the squared error:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{y}_k} E^2 &= \frac{\partial}{\partial \boldsymbol{y}_k}(r - v_k y_k)^2 \\ &= -2E\boldsymbol{v}_k \end{aligned} \tag{3.24}$$

And deriving the update equation for the item feature:

$$\begin{aligned} \boldsymbol{y}_k^{new} &= \boldsymbol{y}_k^{old} - \lambda \frac{\partial}{\partial \boldsymbol{y}_k} E^2 \\ &= \boldsymbol{y}_k^{old} + \lambda E \boldsymbol{v}_k \end{aligned} \tag{3.25}$$

By iterating between the user and item updates until convergence for each $k \in K$ the user- and item-feature matrices, $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{Y}}$ can be estimated without ever having to explicitly handle the feedback matrix. This approach scales with the number of observed items in the feedback matrix, as opposed to the PureSVD approach which scaled with respect to the product of the number of users and items.

When used on a dense matrix, the gradient descent and SVD approaches will produce identical feature matrices. This leads to the same overfitting problems as with PureSVD. An advantage of this approach is that regularisation can be added to the updates to try to address the overfitting.

SVD is a linear approach to predicting, but some non-linearity can be added by making the prediction a function of some non-linear function $G()$:

$$\hat{r} = \sum_{k=1}^{K} G(v_k y_k) \tag{3.26}$$

An example that was found to be useful is a clipping function that discards predictions outside the rating range from any feature (Funk, 2006). For example, for a 1 to 5 rating scale, such a clipping function would be as follows for:

$$G(v_k y_k) = \begin{cases} 0 & x < 0 \\ 5 & x > 5 \\ x & otherwise \end{cases} \tag{3.27}$$

This helps to keep the prediction in the required range.

Other options are non-linear functions such as a sigmoid function like the logistic function

$$G(x) = \frac{1}{1 + e^{-x}} \tag{3.28}$$

For this case, the gradient of $G(x)$ would need to be included in the gradient descent updates. This illustrates the main advantage of the approximate approach over PureSVD in that the model can be altered. As long as the derivatives of the error can be taken, new gradient updates can be calculated to learn other model parameters. A simple example would be to add some user-offset when predicting ratings to account for users using the rating scale differently. The model then predicts a rating as the feature dot product plus the user offset μ_u.

$$\hat{r}_{u,i} = \mu_u + \sum_{k=1}^{K} v_{u,k} y_{i,k} \tag{3.29}$$

3.5.3 Extensions and Evolutions

Other approaches to matrix factorization included probabilistic approaches that seek to address the lack of statistical backing such as probabilistic matrix factorization (PMF) (Mnih & Salakhutdinov, 2008). SVD++ is a generalized approach that allows for bias features and neighbourhood information to be incorporated into the matrix factorisation model (Chen *et al.*, 2011). Asymmetric-SVD incorporates neighbourhood models and latent factor models (Koren, 2008). SVD++ and asymmetric-SVD perform very well on forced-prediction and typically get very good RMSE scores. Interestingly, PureSVD

performs better than these extensions and more complex matrix factorisation approaches for top-N recommendation (Cremonesi *et al.*, 2010).

3.6 Latent Semantic Analysis

The matrix factorisation model has also been used in information retrieval for topic modelling. Topic models are used to model collections of text and other discrete data. Topic models seek to uncover the latent semantic structure of documents (Blei *&* Lafferty, 2009). By discovering complex patterns of words, these topic models uncover the thematic structure of text corpora by describing documents in terms of abstract topics. A topic is a collection of co-occurring words that are semantically related in a given context. Topic models have been used to successfully model collections of many text collections including scientific literature (Wang *&* Blei, 2011; Blei *et al.*, 2003) and newspaper archives (Wei *&* Croft, 2006). They are also not limited to text and can be used to find structure in other large collections of discrete data including genetic data, images, and social networks (Blei, 2012).

Latent Semantic Analysis (LSA) was an early topic-modelling approach that used matrix factorisation to find low-dimensional representations of documents, with LSA documents represented in a document-word co-occurrence matrix. Documents were represented as rows and the corpus vocabulary as columns, with the values being the word counts for each document. This mirrors the user-item feedback matrix from recommender systems, except users are documents and item feedback counts are word counts. Similarly to recommendation, the document-term matrix is typically extremely sparse and the distribution of words uneven, where some words are very common and most words occur rarely. An important concept from topic modelling that is relevant to recommendation is polysemy. Polysemy is where a word can have multiple meanings in different contexts. This is analogous to recommendation where an item may imply different interests based on its context.

3.7 Probabilistic Latent Semantic Analysis

The LSA model struggled to fully capture polysemous and semantically related words for topic modelling. This, coupled with the lack of a theoretical foundation for the model, led to the development of a probabilistic variant of LSA. Probabilistic Latent Semantic Analysis (PLSA) is a statistical alternative to LSA sometimes known as the *aspect model* or *latent aspect model*. The PLSA approach originally given by Hofmann (1999) was a significant step forward for topic modelling. The PLSA model defines a statistical generative model for discrete co-occurrence data that gives a theoretical base that helps explain why these latent factor/aspect models work. Hofmann (2004) later also adapted and discussed the PLSA model for collaborative filtering and showed that it could outperform neighbourhood methods for rating prediction, but it did not see widespread adoption for recommendation tasks.

3.7.1 PLSA model

PLSA is a statistical mixture model that models co-occurrence data in terms of K latent aspects. Aspects are overlapping item clusters that act as user communities (Hofmann *&* Puzicha, 1999); they are defined as multinomial distributions over the items in the system. Thematically or contextually related items should share high probabilities in an aspect in the same way that they were closely positioned in features space for matrix factorisation.

The matrix factorisation model described users in terms of a K-dimensional vector of abstract features; the PLSA model instead restricts the user vector to be a probability distribution. The user is now mapped as a multinomial distribution (mixture) over aspects, which describes how likely they are to select items from a specific aspect. A user can have varying preferences toward many aspects; this differs from Bayesian mixture models where users are assigned to a single item mixture (Breese *et al.*, 1998; Jin *et al.*, 2006). The model then associates an unobserved aspect membership variable,$z \in \{z_1, ..., z_K\}$, with each observation i.e. a $(user \times item)$ pair.

The full generative process is as follows:

1. Select a user u with probability $\mathrm{p}(u)$.
2. Select an aspect z with probability $\mathrm{p}(z \mid u)$.
3. Select an item i with probability $\mathrm{p}(i \mid z)$.
4. Yield (u, i) a user-item pair.

PLSA introduces a conditional independence assumption whereby the user (u) and item (i) are independent given the state of the latent aspect membership assignment z, as can be seen in the graphical model for PLSA shown in 3.4.

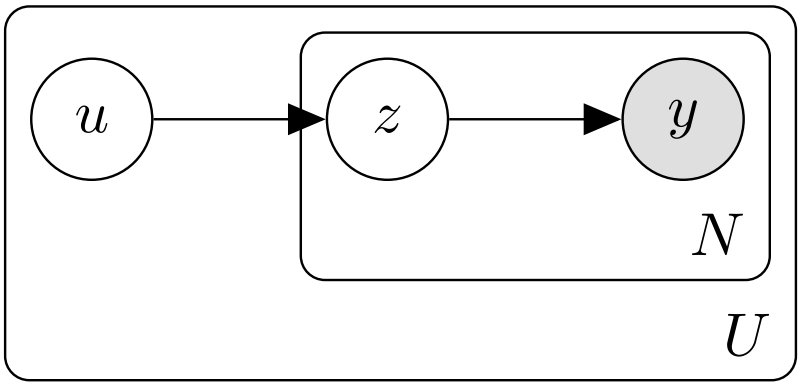


Figure 3.4: Graphical model for asymmetric PLSA.

The generative process results in a $\langle user, item \rangle$ observation pair, where the (latent) aspect membership variable, z, has been discarded. The model has a joint probability over users and items $\mathrm{p}(u, i)$:

$$\mathrm{p}(u, i) = \mathrm{p}(u)\,\mathrm{p}(i \mid u) \tag{3.30}$$

$$\mathrm{p}(i \mid u) = \sum_{z \epsilon Z} \mathrm{p}(i \mid z)\,\mathrm{p}(z \mid u) \tag{3.31}$$

combining (3.30) and (3.31) to get:

$$\mathrm{p}(u, i) = \mathrm{p}(u) \sum_{z \epsilon Z} \mathrm{p}(i \mid z)\,\mathrm{p}(z \mid u) \tag{3.32}$$

The key to understanding the PLSA model is the membership variable z. We can regard the associated z for an observation (u, i) as providing responsibility. The number of aspects is much smaller than the number of users so that

z acts like a bottleneck forcing the model to cluster items and group users into communities. This is similar to restricting the rank with the SVD; there we hoped that the features associated with the largest singular values would capture patterns in the observed data. PLSA models this assumption explicitly as: this observation was observed because user u chose aspect z which delivered item i. The aspects ($\mathrm{p}(i \mid z)$ themselves are distributions over the items. The aspect distributions capture the co-occurrence patterns in the data and form overlapping clusters of items. Items can be associated with multiple aspects and in this way the model can capture context. Users are not assigned to a single aspect but instead are characterized as a mixture over aspects, $\mathrm{p}(z \mid u)$.

It can be helpful to imagine aspects as analogous to genres or specific genre combinations; a user is then characterized by their affinity toward the genres, i.e. the probability they will select a given genre (aspect). The genres are then collections of items characterized by the probability that selecting that genre will yield that item. The PLSA model automatically discovers these 'genres' in the form of aspects, and models the user on how likely they are to select from one.

3.7.2 Relationship with SVD

We can get an equivalent form of the joint probability by reparameterising the model so that items and users are symmetrical with respect to the aspect assignments.

$$\mathrm{p}(u, i) = \sum_{z \in Z} \mathrm{p}(u \mid z)\, \mathrm{p}(z)(i \mid z) \tag{3.33}$$

With this *symmetric* form of PLSA it is easier to see the link to the matrix factorisation model. By placing equation 3.33 in matrix notation, we get equation 3.34

$$\mathbf{P} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T \tag{3.34}$$

where

$$\hat{\mathbf{U}} = (\mathrm{p}(u_i \mid z_k))_{i,k} \tag{3.35}$$

and

$$\hat{\mathbf{\Sigma}} = \mathrm{diag}(\mathrm{p}(z_k))_k \tag{3.36}$$

and

$$\hat{\mathbf{V}} = (\mathrm{p}(i_j \mid z_k))_{j,k}. \tag{3.37}$$

Here we see that the mixture proportions take the form of the singular values, and the user and item mixtures take the form of the feature matrices. The PLSA model is a probabilistic equivalent model to the matrix factorisation model. The models differ in a few key respects, most importantly the objective function. Matrix factorisation minimised the Frobenius (L_2) norm between the true and predicted rating matrix, which is equivalent to minimising the square error between the two. By contrast PLSA relies on the likelihood function and aims to maximise the predictive power of the model (Hofmann, 1999). This causes the joint probability model $\mathbf{P}$ to differ from the matrix factorisation model $\hat{\mathbf{R}}$, in that $\mathbf{P}$ is a now valid probability distribution with a well defined meaning. The elements of $\mathbf{P}$ give the probability that a user would choose an item. The elements of $\hat{\mathbf{R}}$ have no specific meaning and can even be negative.

The probabilistic framework given has many advantages. The first is that the recommendations are explainable. When recommending items to a user from an aspect, the model knows which other items contributed to the user-begin part of that aspect via the aspect assignments. In this way the recommendation can be accompanied by an explanation of which other items caused this item to be recommended to the user. Explainability is an important aspect of practical recommendation (Herlocker *et al.*, 2000), but the probabilistic approach can also address other concerns such as the diversity of the recommendations. The matrix factorisation approach is limited to generating the ranked list of recommendations by sorting on predicted ratings. The probabilistic approach allows for more flexibility in how the recommendation list is generated. For example the recommendations can be generated by sampling from the distributions giving a fresh list of recommendations each time the list is generated.

Another advantage is that the aspects are interpretable, unlike the abstract features of the matrix factorisation model. This makes the model far easier to

troubleshoot by manually inspecting the aspects. It enables data mining and the discovery of user communities and item clusters. Finally, other models can leverage the information in the aspects to improve their recommendations.

3.7.3 Recommendations

PLSA is well suited to item ranking but cannot be directly evaluated via rating prediction. However, the model can be easily extended to handle rating prediction if necessary. A simple example would be to exploit our knowledge of the parameters and predict the rating given the chosen item and the aspect it was assigned to:

$$\mathrm{p}(r \mid y, z) \tag{3.38}$$

This is an interesting case because the predicted rating does not directly depend on the given user. The choice of aspect assignment is dependant on the user's history, however. In this way the predicted rating depends on the context the item finds itself in. Other extensions are given and rating prediction performance results are given by Hofmann (2004), who also shows PLSA to perform 6% better than a neighbourhood model with respect to rating error.

Interestingly, PLSA has inconsistent results when evaluated with item prediction and does not always outperform PureSVD. Barbieri *&* Manco (2011) conclude:

> The PLSA model also seems to suffer from the same overfitting issues, as it is not able to reach the performances of the Pure-SVD. On the other side, if user satisfaction is not taken into account, the PLSA outperforms the Pure-SVD.

One notable use of PLSA in an industry recommender system is at Google News (Das *et al.*, 2007), which uses Probabilistic Latent Semantic Indexing, a document indexing technique based on PLSA, to recommend news articles.

3.7.4 Limitations

PLSA has two major drawbacks: The first is that the PLSA model is only defined for users in the training set. It is not a generative model for new users;

$\mathrm{p}(u)$ is only for users u that have already been seen. The second is that the number of model parameters grows linearly with the number of users, which makes PLSA prone to overfitting. An advantage of the PureSVD approach over the PLSA model is that regularisation can be added to help address overfitting.

3.8 Conclusion

In this chapter we covered the two core approaches to recommendation, namely the memory-based neighbourhood methods and the model-based matrix factorisation models. In addition we covered probabilistic latent semantic analysis, which provides a statistical equivalent model to matrix factorisation. PLSA attempts to provide some theoretical backing to why matrix factorisation models work. PLSA also enables us to provide some meaning behind the mapping to the latent space.

Chapter 4

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generalisation of PLSA that is fully Bayesian. It seeks to address the limitations of the PLSA model, namely the tendency to overfit and the fact that PLSA is not generative with respect to users. This was first proposed for topic modelling by Blei *et al.* (2003). Sparse symmetric Dirichlet priors are placed on the user-aspect mixtures and on the aspect distributions. The Dirichlet is a distribution over multinomial or categorical distributions, making it ideal as a prior for mixtures. It has the added advantage of being conjugate to multinomial and categorical distributions. LDA is sometimes know as the Generative Aspect Model (GAM) due to the fact that it is a fully generative version of the aspect model (PLSA).

4.1 Model

Consider the following joint distribution with the corresponding graphical model given in Figure 4.1.

Firstly we define our ensemble of random variables (that can be inferred) as $X = \{\boldsymbol{\beta}_{1:K}, \boldsymbol{\theta}_{1:U}, \boldsymbol{z}_{1:U}, \boldsymbol{i}_{1:U}\}$, and our ensemble of fixed (hyper) parameters as $Y = \{\boldsymbol{\alpha}, \boldsymbol{\eta}\}$, where all symbols are defined as follows:

$$\mathrm{p}(X \mid Y) = \prod_{k=1}^{K} \mathrm{p}(\boldsymbol{\beta}_k \mid \boldsymbol{\eta}) \prod_{u=1}^{U} \left(\mathrm{p}(\boldsymbol{\theta}_u \mid \boldsymbol{\alpha}) \prod_{n=1}^{N_u} \mathrm{p}(z_{u,n} \mid \boldsymbol{\theta}_u)\, \mathrm{p}(i_{u,n} \mid \boldsymbol{\beta}_{1:k}, z_{u,n}) \right). \tag{4.1}$$

Given this joint distribution and the following graphical model we will first give a summary of each variable before exploring their meaning. After which we show the generative process and give examples of how LDA is used.

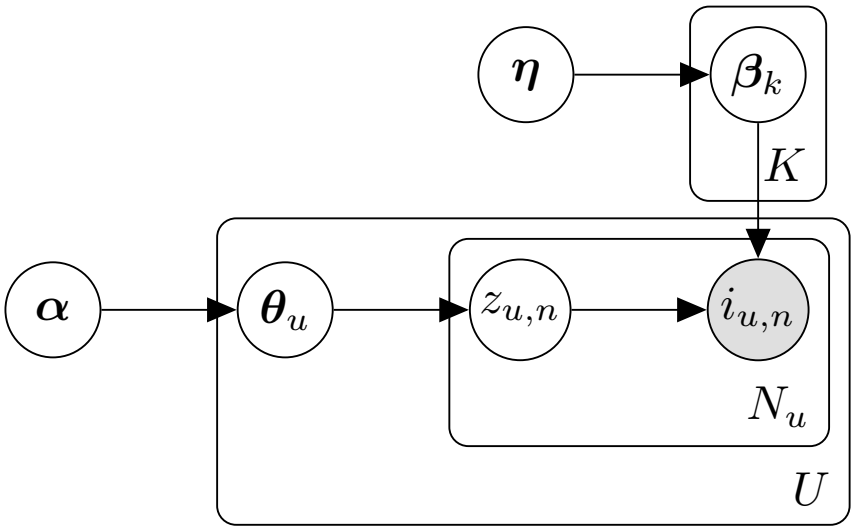


Figure 4.1: Graphical model of the LDA model for recommendation tasks. Note the nested plates of the N observed items for the U users. Each observed item has an associated aspect assignment $z_{u,n}$, which is drawn from the user's aspect distribution $\boldsymbol{\theta}$.

4.1.1 Constants

Constant	Description
U	Number of users.
I	Number of items.
K	Number of aspects.
N_u	Number of observed items for user u.

4.1.2 Notation

Symbol	Dimension	Distribution	Description
$\boldsymbol{\eta}$	I	$\delta(\lvert\boldsymbol{\eta}' - \boldsymbol{\eta}\rvert_2)$	Hyperparameter for the Dirichlet prior to the aspect item distributions.
$\boldsymbol{\alpha}$	K	$\delta(\lvert\boldsymbol{\alpha}' - \boldsymbol{\alpha}\rvert_2)$	Hyperparameter for the Dirichlet prior to the user aspect mixtures.
$\boldsymbol{\beta}$	$K \times I$	$\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\eta})$	Aspect distribution parameters. Each aspect is a multinomial distribution over all items.
$\boldsymbol{\theta}$	$U \times K$	$\boldsymbol{\theta}_u \sim \text{Dir}(\boldsymbol{\alpha})$	Aspect mixture parameters for each user. Each user u is characterised by a multinomial mixture over the K aspects with parameters $\boldsymbol{\theta}_u$.
$\boldsymbol{z}_u$	N_u	$z_{u,n} \sim \text{Multi}(\boldsymbol{\theta}_u)$	Vector of the aspect assignments for each observed item for the user u.
$\boldsymbol{I}_u$	N_u	$i_{u,n} \sim \text{Multi}(\boldsymbol{\beta}_{z_{u,n}})$	Vector of observed items for user u.

As with PLSA the LDA model describes U users and I items in terms of K latent aspects. Each (one of K) aspects is an I-dimensional vector, which is represented by a Multinomial distribution with parameters $\boldsymbol{\beta}_{1:K}$. Unlike PLSA, LDA models the process by which the aspects are first generated. To do so theLDA model places a Dirichlet prior on the aspects; whereby each aspect is drawn from a symmetric Dirichlet with hyperparameter $\boldsymbol{\eta}$:

$$\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\eta}).$$

An aspect is a distribution over items and the modes of the aspects identify co-occuring patterns of items Blei (2012). In doing so aspects can act as communities by describing a common interest pattern between users.

Similar to PLSA, users are described in terms of aspect proportions, the aspect mixture for user u is described by the K-dimensional vector $\boldsymbol{\theta}_u$. The aspect mixture for a user describes how each user belongs to the communities expressed by the aspects. Each item for a user is assigned to an aspect with A latent aspect assignment variable z. So that for each of N observed items for a user $i_{u,n}$, the observed item is assigned to an aspect $z_{u,n}$. The aspect assignments are drawn with the user mixtures; where for the n-th item for user u:

$$z_{u,n} \sim \text{Multi}(\boldsymbol{\theta}_u).$$

Much like the aspects, the user mixtures are assumed to be drawn from a Dirichlet prior with a symmetric hyperparameter. The prior has the hyperparameter $\boldsymbol{\alpha}$ and is used to generate the mixtures; for user u the user's aspect mixture is generated from the prior as such:

$$\boldsymbol{\theta}_u \sim \text{Dir}(\boldsymbol{\alpha}).$$

From the graphical model we can now summarise our generative process as follows:

1. Generate each aspect of the K aspects: $\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\eta})$.
2. For each user $u \in U$:
    a) Choose the number of items for the user: N_u.
    b) Select the user's aspect mixture: $\boldsymbol{\theta}_u \sim \text{Dir}(\boldsymbol{\alpha})$.
    c) For each of the N_u items the user u chooses $i_{u,n}$:
        i. Select an aspect assignment from the user's mixture: $z_{u,n} \sim \text{Multi}(\boldsymbol{\theta}_u)$.
        ii. Select an item from the assigned aspect $i_{u,n} \sim \text{Multi}(\boldsymbol{\beta}_{k=z_{u,n}})$.

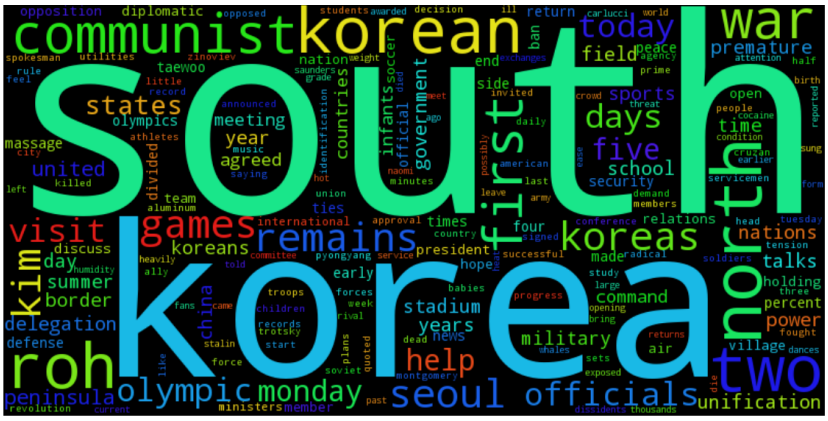
4.2 Examples

4.2.1 Topic modelling

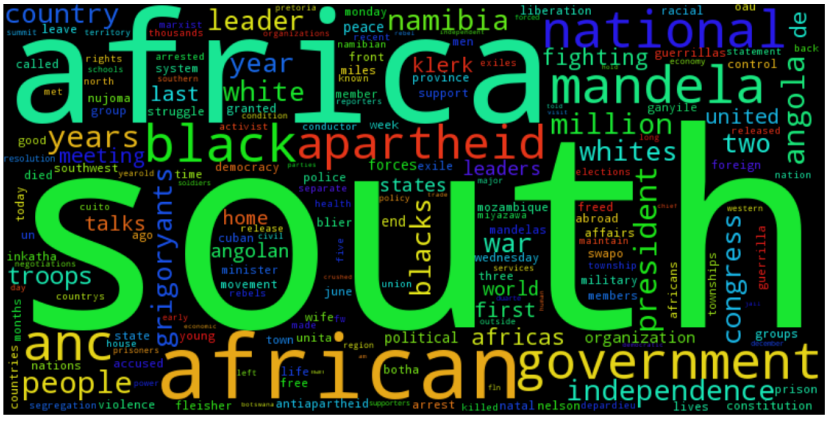
LDA originally came from topic modelling and was used to model large collections of text in terms of latent topics. When used for topic modelling the aspects form topics, which are collections of thematically related words. If we map the recommender system terminology back to topic models we get the following for LDA:

- Items are words.
- Users are documents.
- Aspects are topics.

Each document is now characterized by its membership to the topics it exhibits (its mixture over topics). This captures the intuition that a document can be about varying topics in different proportions. Topics can also overlap so that one word can belong to multiple topics which captures the idea that a word can have different meanings in different contexts.



(a) A topic where *south* and *korea* are the most likely words. We see words related to the 1988 Olympics held in South Korea and words related to the conflict with North Korea.



(b) A topic where *south* and *africa* are the most likely words. Despite the fact that 'south' co-occurs with the 'south korea' topic, the other related terms are immediately recognizable as being related to South Africa, seemingly focused on coverage of apartheid. The two topics overlap slightly with the most likely word but otherwise are distinct and, from a human perspective, intuitive.

Figure 4.2: Word clouds of topic distributions over words. Size of the word is proportional to the probability of the word being selected for the topic; larger words are more likely in the topic.

As an example consider Figure 4.2 which shows two topics from an LDA with 50 topics trained on the Associated Press dataset. The Associated press

dataset is a corpus of Newswire articles from between the years 1988 and 1990.

Each word cloud represents the aspect distribution and the size of the word indicates its relative probability (bigger word = higher probability). We can see that the topics overlap on their most probable word 'south', but the rest of the significant words differentiate the topics between being related to 'South Africa' or 'South Korea'.

With the LDA model a document is considered simply as a collection of word counts. This is known as the *bag of words* assumption. The relative order of the words does not matter, only that they occurred in a document. A corpus can the be represented as a document-word matrix that has documents as rows and the vocabulary as the columns. A document is then a mostly zero vector with counts for the words that did occur.

4.2.2 Movie recommendations

We once again look at using for the LDA model for recommendation.Let us consider the example of recommending movies to users based on the movies that they have watched previously. In much the same way as the document-term matrix from topic modelling we can construct a user-movie matrix. The user-movie matrix is the implicit feedback matrix that we are familiar with from recommender systems that has the interaction counts for users. In this case our interaction is watching a movie.

Given the observed interactions we are interesting in the item distributions and the user mixtures. The top items from four example aspects are shown in Figure 4.3. Note how each aspect is analogous to a genre or style of movie. With that interpretation a user's aspect mixture describes their proportional interest in various genres. A recommendation then involves recommending further movies from genres(aspects) that the user is interested in.

Consider two basic approaches to recommending items with LDA. The first is to recommend the most likely items for a given user. This corresponds to

sorting the item probabilities given a user's aspect mixture:

$$\mathrm{p}(i \mid \boldsymbol{\theta}_u, \boldsymbol{\beta}) \propto \sum_{k=1}^{K} \boldsymbol{\beta}_k \cdot \theta_{u,k}. \tag{4.2}$$

The second approach is to sample items based on the generative model. If we wish to generate N_u^{new} items for user u we can:

1. For n' in N_u^{new}:

   a) Select an aspect $z_{u,n'} \sim \mathrm{Multi}(\boldsymbol{\theta}_u)$.

   b) Recommend item $i_{u,n'}$ selected as $i_{u,n'} \sim \mathrm{Multi}(\boldsymbol{\beta}_{z_{u,n'}})$.

This generates a random list of items each time it is run. In situations were more diversity in recommendation is desired, this may be a preferable approach. Much like neighbourhood models, the recommendations are explainable as 'users similar to you enjoyed'.

LDA gives a system designer a great deal of flexibility in terms of how the model distributions are used for recommendation. If one wishes to perform item-based recommendation, i.e. recommend similar items to an item a user has selected, it is simple to do so with the LDA, if we wish to recommend items for user u based on their interaction with item $i_{u,n}$:

1. With the aspect $z_{u,n}$ that the model assigned to the interaction of the user and item.

2. For n' in N_u^{new}:

   a) Recommend item $i_{u,n'}$ selected as $i_{u,n'} \sim \mathrm{Multi}(\boldsymbol{\beta}_{z_{u,n}})$.

Because the assigned aspect $z_{u,n}$ is dependent on the user, the listed items will be personalised to them and related by the aspect to the item in question. This highlights the flexibility in generating a recommendation list from a probabilistic framework as opposed to the previous deterministic ones. In general a probabilistic approach is preferred for the flexibility it gives the system designer, along with the insight the model can give on users and items (Barbieri & Manco, 2011).

Aspect	
3	**9**
Jurassic Park	Citizen Kane
Men in Black	One Flew Over the Cuckoo's Nest
Matrix, The	Godfather, The
Rock, The	Midnight Cowboy
Fugitive, The	Graduate, The
29	**15**
Beauty and the Beast	Raiders of the Lost Ark
Lion King, The	Star Wars: Episode V - The Empire Strikes Back
Aladdin	Star Wars: Episode IV - A New Hope
Snow White	Star Wars: Episode VI - Return of the Jedi
Cinderella	Indiana Jones and the Last Crusade

Figure 4.3: Top 5 most likely items from 4 aspects taken from an LDA model trained with 50 aspects on the Movielens 1 Million Dataset. Much like the news topics, these aspects take on easily recognisable themes. It is worth noting that the LDA has no additional information about these movies apart from knowing which users rated them. Yet the clusters of items (movies in this case) are human interpretable - not just interpretable in the sense that we know that the number associated with an item is a probability (this is not the case with matrix factorisation where the feature weights have no interpretation), but also in the sense that a human can interpret that aspect 29 is a cluster of animated childrens' movies.

An important feature when using LDA for recommendation is that the aspects are interpretable. LDA embeds users in an aspect space; similarly, matrix factorisation performs a linear embedding of users and items to a feature space. However the matrix factorisation features are not interpretable and can take any numerical value, even a negative one, but if one were to inspect the feature weight for a user, it has no meaning. The LDA embeddings however are probabilities and we understand that the value indicates the likelihood that the user will select an aspect. This gives the LDA additional flexibility: the user mixtures and aspect mixtures can be used not just for recommendation, but also for extending the model or enhancing other tasks such as dataset exploration, e.g. discovering user communities or item clusters.

An additional feature of the LDA model is that the aspects often make in-

tuitive sense and are human readable. Considering the word clouds and movie aspects shown previously, we see that both produced sensible outputs that are highly interpretable.

4.2.3 Priors

To understand how LDA works and why it forms these clusters of co-occurring items, we can consider the joint distribution. The joint is proportional to the posterior, which can be factorising as in equation 4.3

$$\mathrm{p}(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{z} \mid i, \boldsymbol{\alpha}, \boldsymbol{\eta}) \propto \mathrm{p}(i \mid \boldsymbol{z}, \boldsymbol{\beta})\, \mathrm{p}(\boldsymbol{z} \mid \boldsymbol{\theta})\, \mathrm{p}(\boldsymbol{\theta} \mid \boldsymbol{\alpha})\, \mathrm{p}(\boldsymbol{\beta} \mid \boldsymbol{\eta}) \tag{4.3}$$

The model wants to place high probabilities on the observed users and maximise equation 4.3. Each of the factors in the product are probability distributions. If we consider a probability distribution as a vector $\mathbf{p}$ where $\sum \mathbf{p} = 1$ and each element is in the range $0 \leq p \leq 1$, the product of two such vectors is maximised when as few elements as possible are non-zero in both. This implies that the joint probability is maximised when each of the conditional distributions in equation 4.3 is as sparse as possible. Considering this we can examine how this affects each one individually:

$\mathrm{p}(\boldsymbol{z} \mid \boldsymbol{\theta})$: The model is penalised for placing probability mass in too many aspects in $\boldsymbol{\theta}$, so that the probability of choosing an aspect z is large. In other words, ideally a user should strongly belong to few aspects rather than weakly to many.

$\mathrm{p}(i \mid \boldsymbol{z}, \boldsymbol{\beta})$: Similarly, the aspect distributions $\boldsymbol{\beta}$ should form sparse segregated item clusters.

$\mathrm{p}(\boldsymbol{\beta} \mid \boldsymbol{\eta})$: A sparse Dirichlet prior will give high probability to aspect distributions that are sparse.

These factors interact to encourage $\boldsymbol{\beta}$ and the user θ variables to be sparse. The priors do not actually force the variables to be sparse, and sufficient evidence can overwhelm their influence (see the example in section 2.4 which illustrates the influence of priors). The $\mathrm{p}(i \mid \boldsymbol{z}, \boldsymbol{\beta})$ terms encourages having few items with meaningful probability in an aspect; this encourages finding

segregated clusters of items. $p(\boldsymbol{z} \mid \boldsymbol{\theta})$ gives a high posterior probability when $\boldsymbol{\theta}$ has few aspects per user, thereby penalizing the user for having diffuse membership over many aspects. If the model should assign a user to few aspects and aspects consist of few items, the optimal solution is to find collections of co-occurring items that best generalise the observed data (Reed, 2012).

In a more general sense the prior parameters give the system designer control over the type of mixtures desired for the user and item-aspect mixtures. The prior parameter $\boldsymbol{\eta}$ plays an important role in smoothing $\boldsymbol{\beta}$ (Blei *et al.*, 2003). The LDA model is encouraged to place probability mass in β on few items. In recommender systems where most of the ratings will be for few items, this may cause the LDA to ignore the majority of items and instead favour popular items. The hyperparameter $\boldsymbol{\eta}$ can be used to encourage LDA to do the opposite. By making the $Dirichlet(\boldsymbol{\eta})$ dense as opposed to sparse, the model is more inclined to spread the probability mass over more items and thus hopefully improve the coverage of the possible items.

The priors are a major advantage for a system designer using LDA for recommendation, as they offer a greater degree of freedom when specifying the model. Instead of just having control of the rank of the representation K, the designer can encourage different levels of sparsity in the representations found.

4.3 Extensions to Latent Dirichlet Allocation

4.3.1 Rating Prediction

Both aspect models that we have looked at do not model any associated ratings with the co-occurrence pairs. One of the strengths of these Bayesian models is that they are easily extended and can be quickly adapted to free- and forced-prediction with ratings.

For rating prediction we are interested in predicting the ratings r associated with a $\langle user - item \rangle$ pair. To do so, a rating variable r can be added and we are then interested in the conditional $p(r \mid u, y)$ Hofmann (2004). This ap-

proach does not play to the strengths of aspect models and it is more intuitive to consider the role of user choice.

For free prediction the generative aspect model can be extended into a two-stage process:

1. The user selects an item (modelled naturally by the aspect model)
2. (Optionally) The user supplies a rating for the item

One approach is to use the ordered-logit model, which is an econometric model (Rubin & Steyvers, 2009). This approach introduces three additional model parameters for the ordered-logit model addition, but the two-stage process means that inference is still straightforward. Another approach is the Uses-Rating-Profile (URP) model that builds on the LDA (Marlin, 2004). However, URP performs worse than LDA for ranking (although it is better for ratings prediction (Barbieri & Manco, 2011)).

An advantage of two-stage style models is that they expose more rich semantics to enable recommendation. An interesting approach is to recommend items that have high expected ratings but a low probability of occurring for a user. The system will then recommend an item it expects the user to enjoy but was unlikely to select themselves, thereby hopefully increasing the novelty of the recommendation.

4.3.2 Incorporating LDA into other models

As a Bayesian model the Generative Aspect Model is powerful in that it can be incorporated into more complex Bayesian models for specific domains. Exploiting the GAM's ability to get a low-dimensional embedding of unstructured sparse data is one such use. An example is for recommending text. The GAM can be used to represent the text as aspect mixtures, and the extended system can use the aspect mixtures to generate recommendations. This can be used to recommend scientific articles (Wang & Blei, 2011) where the GAM was trained on article abstracts and recommendations were made to users based on their interest in aspects. A similar approach is used at the *New York Times*

to recommend news articles (Spangher, 2015).

In general the GAM can be used to give a fixed representation of unstructured metadata (Agarwal & Chen, 2010).

4.3.3 Domain Flexibility: Volatile Items

An interesting use of the latent aspect model and an example of its flexibility is embedding users and items in the same latent space given a common third layer. In this case it was the use of $< user - item - search\, query >$ triadic data as opposed to the usual dyadic data we have considered thus far. This was for an online auction site, so items were short-lived compared to users. In this situation most of the items were new and did not overlap with the user's current history. The approach taken was to train the modal with $user - search\, qeury$ and $item - search\, query$ co-occurence data at once. Users and items were now both embedded in the latent aspect space, and aspects were distributed over search queries. New items could be added based on the search queries that would return them, and users' interest was modelled in terms of what they searched for as opposed to the items they viewed/bought (Ovsjanikov & Chen, 2010).

This is a good example of the flexibility of the latent aspect model and how the lack of structure it imposes on the input data allows for many interesting approaches to recommendation.

4.3.4 LDA extensions from other fields

There exist many extensions and evolutions of LDA from topic modelling. There are versions that add dependency between the aspects. There are models that construct trees of aspects, where aspects go from general to specific focus as the depth of the tree increases. Other models relax some of the assumptions made. The models discussed thus far all retain the core exchangeability assumption that users and items pairs are exchangeable. Additionally the models required that the number of aspects K be fixed and known in ad-

vance. We can relax these assumptions if necessary using a variety of Bayesian non-parametrics (Blei, 2012).

4.4 Conclusion

LDA generalises PLSA and by extension provides a fully Bayesian interpretation of the matrix factorisation model.

LDA conceptually also has advantages over matrix factorisation. The recommendation strategy is flexible and enables the recommendation list to be generated stochastically to improve diversity. The parameters of the model are interpretable, which aids debugging, but it is also useful for integrating LDA with other algorithms and models. Hybrid recommenders are extremely common in industry (Gomez-Uribe & Hunt, 2015), and the ability for the LDA parameters to be consumed by other models is very attractive.

LDA is well studied in the context of topic modelling, but has not received the extensive coverage that neighbourhood methods and matrix factorisation methods have enjoyed in recommender systems. Therefore, even though there are many evolutions of LDA that exist it is not certain when to use them or even in which contexts LDA itself should be used.

Chapter 5

Approximate Bayesian Inference

5.1 Introduction

For a Bayesian model with latent variables $\boldsymbol{\zeta}$ over some data $\boldsymbol{x}$, the model defines a joint probability distribution $\mathcal{P}$ over its variables:

$$\mathrm{p}(\boldsymbol{x}, \boldsymbol{\zeta}) = \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{\zeta}). \tag{5.1}$$

After observing some data $\boldsymbol{x}$, Bayesian inference involves computing the posterior distribution conditioned on the observed data

$$\mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x}) = \frac{\mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{\zeta})}{\int \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta}')\,\mathrm{p}(\boldsymbol{\zeta}')d\boldsymbol{\zeta}'} \tag{5.2}$$

or we may wish to calculate some statistic of the posterior distribution; any statistic can be expressed in terms of an expectation of the posterior as function of $\boldsymbol{\zeta}$, $f(\boldsymbol{\zeta})$

$$\mathbb{E}\left[f(\boldsymbol{\zeta})|\boldsymbol{x}\right] = \frac{\int f(\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{\zeta})d\boldsymbol{\zeta}}{\int \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta}')\,\mathrm{p}(\boldsymbol{\zeta}')d\boldsymbol{\zeta}'}. \tag{5.3}$$

Computing the posterior and posterior expectations both involve integrals that quickly become computationally intractable for large models. The marginal likelihood in the denominator of Bayes' theorem involves integrating over all the possible hidden states, and this computation grows exponentially with model complexity. For complex models with large datasets we therefore require methods to approximate these integrals or to approximate the posterior directly.

As a relevant example, consider the LDA model. We wish to compute the posterior distribution of the latent variables given the priors and observed interaction counts $\boldsymbol{i}$:

$$\mathrm{p}(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta} | \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\eta}). \tag{5.4}$$

The aspect assignments $\boldsymbol{z}_u$ and their proportions $\boldsymbol{\theta}_u$ are conjugate and this makes them computationally convenient. However, the coupling between the aspects $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ cause the posterior to become analytically intractable (Blei *et al.*, 2003).

Fortunately there exist many forms of approximate inference that can be considered for Bayesian models. We will consider three different approaches to approximate inference: Gibbs sampling, Variational Inference (VI), and Expectation Propagation (EP). The goal of all three is to find an approximate posterior $q(\boldsymbol{\zeta})$ that is computationally tractable. Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method that uses sampling to approximate integrals, and the approximation comes from limiting the computing time. Variational Inference and Expectation Propagation both find approximate posteriors by minimising the Kullback–Leibler divergence (a measure of difference between distributions) between the approximate and true posterior.

5.2 Markov Chain Monte Carlo: Gibbs Sampling

Markov Chain Monte Carlo (MCMC) methods are primarily concerned with approximating expectations, where $f(x)$ is some function of x, in the form:

$$\mathbb{E}_p\left[f(x)\right] = \int f(x)\,\mathrm{p}(x)dx. \tag{5.5}$$

In terms of posterior inference we are interested in calculating the statistics of the posterior. Any of the statistics can be calculated in terms of posterior expectations:

$$E_{\mathcal{P}}\left[f(\boldsymbol{\zeta})|\boldsymbol{x}\right] = \frac{\int f(\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})d\boldsymbol{\zeta}}{\int \mathrm{p}(\boldsymbol{\zeta})\,\mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})d\boldsymbol{\zeta}}, \tag{5.6}$$

which are intractable in many cases. MCMC methods construct a Markov Chain that has the posterior of interest as its stationary distribution. This

Markov chain can then be used to draw samples from the posterior, and, by using Monte Carlo integration, the samples can be used to evaluate an approximate version of the expectation of interest.

5.2.1 Monte Carlo Integration

Monte Carlo integration is a method of evaluating $\mathbb{E}\left[f(x)\right]_p$ by drawing samples from $\mathrm{p}(x)$. Formally it is based on the idea that any desired expectation can be estimated via ergodic averages (Yildirim, 2012). In other words, any expectation (thus any statistic) of the distribution $\mathrm{p}(x)$ can be evaluated by drawing N simulated samples from the distribution and then approximating the expectation as a sum

$$\mathbb{E}\left[f(x)\right]_p \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n). \tag{5.7}$$

Where x_i is the i-th simulated sample from the distribution $\mathrm{p}(x)$. This is much like estimating the population expectations from the sample expectations, for example the mean of the posterior can be estimated as:

$$\mathbb{E}\left[x\right] \approx \frac{1}{N}\sum_{n=1}^{N} x_n. \tag{5.8}$$

The accuracy of the approximation is based on the number of samples N that are drawn. To get a more accurate approximation more samples can simply be drawn; the limits of the approximation are then based on the computational limit of not being able to draw infinite samples from the posterior. This still leaves the problem of generating N samples from the posterior, as the posterior can be in a non-standard form with no way of easily drawing samples. One way is to construct a Markov chain that has the posterior as its stationary distribution (Gilks *et al.*, 1995).

5.2.2 Markov Chains

For a generated sequence (chain) of random variables $x^{(0)}, x^{(1)}, \ldots$ such that at each time $t \geq 0$ the next state of the variable $x^{(t+1)}$ is sampled from the distribution $\mathrm{p}(x^{(t+1)}|x^{(t)})$ that only depends on the current state of the variable:

$$x^{(t+1)} \sim \mathrm{p}(x^{(t+1)}|x^{(t)}). \tag{5.9}$$

Therefore, given the variable in the current state $x^{(t)}$, the next state does not depend on the history of the chain $x^{(0)}, x^{(1)}, ..., x^{(t-1)}$; this sequence is a Markov chain (Gilks *et al.*, 1995).

As t increases, the samples x^t will increasingly look like samples from some stationary distribution $\pi(x)$ as the chain converges. The chain depends on the initial state x^0 and early samples will not be independent of it. After a sufficient number of samples, the chain 'forgets' its initialisation and $p^{(t)}(x^{(t)}|x^{(0)})$ converges to a stationary distribution (Gilks *et al.*, 1995). The time this takes is known as the *burn-in* period. During the burn-in period the samples may not properly represent the stationary distribution and they are commonly discarded before performing Monte Carlo integration. After the burn-in period the samples in the chain are dependent samples from the stationary distribution (Gilks *et al.*, 1995). It is difficult to know when the chain has burnt in and most methods are based on heuristics (Geyer, 2017).

5.2.3 Metropolis-Hastings Algorithm

The goal now is to construct a Markov chain that has the posterior distribution $\mathcal{P}$ as its stationary distribution π. This is done simply via the Metropolis-Hastings algorithm.

The Metropolis-Hastings algorithm works by first sampling a candidate sample $c^{(t)}$ from a *proposal distribution* $\mathcal{Q}(c|x^{(t)})$ that may depend on the current sample state. The candidate point is then chosen to be accepted as the next sample with the probability $\alpha(x_t, c^{(t)})$, which is defined as:

$$\alpha(x^{(t)}, c^{(t)}) = \min(1, \frac{\pi(c^{(t)})\mathcal{Q}(x^{(t)}|c^{(t)})}{\pi(x^{(t)})\mathcal{Q}(c^{(t)}|x^{(t)})}) \tag{5.10}$$

If the candidate sample is accepted it becomes the next sample state $x^{(t+1)} = c^{(t)}$; otherwise the chain does not change $x^{(t+1)} = x^{(t)}$.

$$x^{(t+1)} = \begin{cases} c^{(t)} & \text{with probability } \alpha(x^{(t)}, c^{(t)}) \\ x^{(t)} & \text{with probability } 1 - \alpha(x^{(t)}, c^{(t)}). \end{cases} \tag{5.11}$$

Algorithm 1 shows the general procedure for general Metropolis-Hastings

Algorithm 1 Metropolis-Hastings algorithm

Initialise x^0
$t \leftarrow 0$
repeat
  Sample a point y from the proposal distribution $c^{(t)} \sim q(c|x^{(t)})$
  Calculate the acceptance probability $\alpha(x^{(t)}, c^{(t)})$
  Accept $c^{(t)}$ with probability $\alpha(x^{(t)}, c^{(t)})$
  if $c^{(t)}$ is accepted **then**
    Set $x^{(t+1)} = c^{(t)}$
  else
    Set $x^{(t+1)} = x^{(t)}$
  end if
until $t \geq$ number of samples required

To get a suitable Markov Chain, we simply need to select a proposal distribution. Gibbs sampling defines a convenient proposal distribution and is a very common MCMC method.

5.2.4 Gibbs Sampling

Consider a vector of random variables $\boldsymbol{\zeta}$, which for example could be the parameters of a Bayesian model. Gibbs sampling provides a proposal distribution for updating the i-th component of $\boldsymbol{\zeta}$ using the full conditional distribution of ζ_i (Gilks *et al.*, 1995). The full conditional distribution is the distribution of the i-th component of $\boldsymbol{\zeta}$ conditioned on all the other components $\boldsymbol{\zeta}_{-i}$:

$$\mathrm{p}(\zeta_i|\boldsymbol{\zeta}_{-i}) = \frac{\mathrm{p}(\boldsymbol{\zeta})}{\int \boldsymbol{\zeta} d\zeta_i}. \tag{5.12}$$

The proposal distribution for the i-th random variable in $\boldsymbol{\zeta}$ is the full conditional for the ζ_i conditioned on the current state of all the components of $\boldsymbol{\zeta}$ excluding ζ_i:

$$\mathcal{Q}(c|\boldsymbol{\zeta}^t_{-i}) = \mathrm{p}(\zeta_i|\boldsymbol{\zeta}^t_{-i}). \tag{5.13}$$

If we substitute 5.13 into 5.10 the acceptance probability is always 1, so that all candidates are always accepted.

A Gibbs sampler then sweeps through each variable and samples from its conditional while the other variables are held constant by considering the state of all the other variables $\boldsymbol{\zeta}^{(t)}_{-i}$ excluding the current ζ_i.

Algorithm 2 Gibbs Sampling

Initialise $\boldsymbol{\zeta}^{(0)}$ to some initial values
for each time period t=1,2... **do**
  for each random variable in $\boldsymbol{\zeta}$: $\zeta_1, \zeta_2, ...$ **do**
    $\zeta_i^{(t)} \sim \mathrm{p}(\zeta_i | \boldsymbol{\zeta}_{-i})$
  end for
end for

After some time the values of $\boldsymbol{\zeta}^t$ at each time period t will be samples from the distributions over the components of $\boldsymbol{\zeta}$.

For Bayesian models we are interested in obtaining samples for the model parameter distributions so that we can estimate the posterior. Fortunately the Bayes Net representation allows us to easily obtain conditional distributions of each parameter given the others. A Gibbs sampler can then simply sample from each conditional in the factorised posterior in turn.

5.2.5 Example Gibbs Sampling

As a simple toy example, consider a model with three dependent parameters: α, β, and θ. Gibbs sampling is performed via the simple procedure in algorithm 3. Note that the updates can be done in any order.

Algorithm 3 Gibbs sampling example for a model with three parameters

Initialise $\alpha^{(0)}$, $\beta^{(0)}$ and $\theta^{(0)}$
for each time step t=1,2... **do**
  $\alpha^{(i)} \sim \mathrm{p}(\alpha | \beta^{(i-1)}, \theta^{(i-1)})$
  $\beta^{(i)} \sim \mathrm{p}(\beta | \alpha^{(i)}, \theta^{(i-1)})$
  $\theta^{(i)} \sim \mathrm{p}(\theta | \alpha^{(i)}, \beta^{(i)})$
end for

5.2.6 Collapsed Gibbs

For collapsed Gibbs inference, one or more of the variables are marginalised out. Sampling from the marginalised distribution is usually tractable when the marginalised distribution is a conjugate prior for the variable being sam-

pled for. Considering the example seen in algorithm 3, if the variable β is marginalised, the Gibbs procedure becomes that in algorithm 4.

Algorithm 4 Collapsed Gibbs sampling example for a model with three parameters with one collapsed out

Initialise $\alpha^{(0)}$ and $\theta^{(0)}$
for iteration i=1,2... **do**
  $\alpha^{(i)} \sim \mathrm{p}(\alpha|, \theta^{(i-1)})$
  $\theta^{(i)} \sim \mathrm{p}(\theta|\alpha^{(i)})$
end for

5.2.7 Fast Collapsed Gibbs for LDA

Porteous *et al.* (2008) derives a faster Gibbs sampler for Latent Dirichlet Allocation that exploits structure in the solution to speed up sampling.

5.2.8 Collapsed Gibbs for Latent Dirichlet Allocation

Determining a Gibbs sampler for the LDA model becomes relatively straightforward by collapsing out most of the variables so that we only sample the aspect assignments for the j-th observed item feedback z_j. Griffiths (2002) developed such a collapsed Gibbs sampler, which is given in summary below.

We wish to sample the aspect assignments $\boldsymbol{z}_u$ given the observed items $\boldsymbol{i}_u$ for a user u. We can get the items from the observed feedback $\boldsymbol{y}$ because each observed feedback $y_n = y_{u,i}$ has an associated user u and item i. The conditional posterior for the n-th observed item's aspect assignment $z_{u,n}$ is given by

$$\mathrm{p}(z_{u,n}|\boldsymbol{z}_{u,-n}, \boldsymbol{y}) \propto \underbrace{\mathrm{p}(i_{u,n}|z_{u,n}, \boldsymbol{z}_{u,-n}, \boldsymbol{i}_{u,-n})}_{\text{(a)}} \underbrace{\mathrm{p}(z_{u,n}|\boldsymbol{z}_{u,-n})}_{\text{(b)}}. \tag{5.14}$$

This is an application of Bayes' rule. It does not depend on the multinomial parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ as they have been marginalised out.

To calculate the first term (a) we integrate out $\boldsymbol{\beta}$:

$$\text{(a): } \mathrm{p}(i_{u,n}|z_{u,n}=k,\boldsymbol{z}_{u,-n},\boldsymbol{i}_{u,-n}) = \int \underbrace{\mathrm{p}(i_{u,n}|z_{u,n}=k,\boldsymbol{\beta}_k)}_{\beta_{k,i_{u,n}}} \underbrace{\mathrm{p}(\boldsymbol{\beta}_k|\boldsymbol{z}_{u,-n},\boldsymbol{i}_{u,-n})}_{\text{a:1}} d\beta_k. \tag{5.15}$$

The first term is $\beta_{k,i_{u,n}}$, which is the item probability for the item $i_{u,n}$ in $y_{u,n}$ and the current aspect k. The second term in (a:1) can be obtained with Bayes' rule once again:

$$\text{(a:1): } \mathrm{p}(\boldsymbol{\beta}_k|\boldsymbol{z}_{u,-n},\boldsymbol{i}_{u,-n}) \propto \mathrm{p}(\boldsymbol{y}_{u,-n}|\boldsymbol{\beta}_k,\boldsymbol{z}_{u,-n}) \underbrace{\mathrm{p}(\beta_k)}_{\mathrm{Dir}(\boldsymbol{\eta})}. \tag{5.16}$$

The right term in (a:1) is a Dirichlet with $\boldsymbol{\eta}$ as a parameter, as $\mathrm{Dir}(\boldsymbol{\eta})$ is the prior on the components of $\boldsymbol{\beta}$. It is conjugate to $\mathrm{p}(\boldsymbol{i}_{u,-i}|\beta_j,\boldsymbol{z}_{u,-i})$. The posterior distribution $\mathrm{p}(\beta_j|\boldsymbol{z}_{u,-i},\boldsymbol{i}_{u,-i})$ can be simplified to a $\mathrm{Dir}(\eta_k + N^{(u)}_{-i,k})$, where $N^{(i)}_{-n,k}$ is the number of times item i is assigned to aspect k, not including the current item. $\boldsymbol{z}_{u,-n}$ acts to partition the items into sets and only the items assigned to aspect j affect β_j

combining all and completing the integral to get an expression for (a). $N_{-n,k}$ is the number of items assigned to aspect j excluding the current item, and $N^{(i)}_{-n,k}$ is the number times item i has been assigned to aspect k, excluding the current item.

$$\text{(a): } \mathrm{p}(i_{u,n}|z_{u,n}=k,\boldsymbol{z}_{u,-n},\boldsymbol{i}_{u,-n}) = \frac{N^{(i)}_{-n,k}+\eta}{N_{-n,k}+I\eta} \tag{5.17}$$

The second term in equation 5.14 can be evaluated in much the same way as the first to get

$$\begin{aligned}\text{(b): } \mathrm{p}(z_{u,n}=k|\boldsymbol{z}_{u,-n}) &= \int \mathrm{p}(z_{u,n}=k|\boldsymbol{\theta}_u)\,\mathrm{p}(\theta_u|\boldsymbol{z}_{u,-n})d\boldsymbol{\theta}_u \\ &= \frac{N^{(u)}_{-n,k}+\alpha_k}{N^{(u)}_{-n}+K\alpha_k}.\end{aligned} \tag{5.18}$$

Combining (a) and (b) for equation 5.14 we get the full conditional

$$\mathrm{p}(z_{u,n}=k|z_{u,-n},\boldsymbol{i}_u) \propto \underbrace{\frac{N^{(i)}_{-n,k}+\eta_k}{N_{-n,k}+I\eta_k}}_{\text{(a)}} \underbrace{\frac{N^{(u)}_{-n,k}+\alpha_k}{N^{(u)}_{-n}+K\alpha_k}}_{\text{(b)}} \tag{5.19}$$

folding all constant terms into the normaliser $B = N_{-n}^{(u)} + K\alpha_k$; $N_{-n}^{(u)}$ is the number of observed for a user excluding the current.

We get the following expression for the sampling conditional where $N_{-n,k}^{(i)}$ is the the total number of times item i is assigned to aspect k excluding the current, $N_{-n,k}^{(u)}$ is the number of items for the current user assigned to aspect k excluding the current item and $n_{-n,k}$ is the total number of items assigned to aspect k excluding the current.

$$\mathrm{p}(z_{u,n} = k|\boldsymbol{z}_{u,-n}, \boldsymbol{i}) = \frac{1}{B}\frac{(N_{-n,k}^{(i)} + \eta_k)(N_{-n,k}^{(u)} + \alpha_k)}{N_{-n,k} + I\eta_k} \tag{5.20}$$

The Gibbs procedure is simple: loop through each observation and sample a new aspect assignment; once the chain has stabilised, sample an aspect assignment for each observation. The estimates are computationally simple, only being dependent on some counts (Porteous *et al.*, 2008).

Given the samples, we can derive estimates for each user's aspect distributions $\boldsymbol{\theta}_u$. The estimate only requires the count of how many items for user u is assigned to aspect k: N_k^u and the total number of items for the current user N^u:

$$\theta_{u,k} = \frac{N_k^u + \alpha}{N^u + K\alpha_k}. \tag{5.21}$$

The aspect distributions can be estimated in a similar way, only relying on the number of times item i is assigned to aspect k: N_k^i and the total number of words assigned to aspect k: N_k.

$$\beta_{i,k} = \frac{N_k^i + \eta_k}{N_k + I\eta_k} \tag{5.22}$$

The inference procedure is then straightforward. The aspect assignments do not need to be recorded, only the count totals do. The Gibbs sampler can then loop through each observation, obtain a new sample, and then update the counts. This is a very memory-efficient procedure.

Algorithm 5 Collapsed Gibbs inference procedure for Latent Dirichlet Allocation

initialise each z_n between 1 and K
while chain is not stationary **do**
  for each observation y_n consisting of user u and item i **do**
    $z_{u,n} \sim \frac{1}{B}\frac{(N^{(i)}_{-n,k}+\eta_k)(N^{(u)}_{-n,k}+\alpha_k)}{N_{-n,k}+I\eta_k}$
    update the counts: $N_{u,k}$, $N_{i,k}$ and N_k
  end for
end while
for each user u **do**
  for each aspect k **do**
    estimate $\theta_{u,k} = \frac{N^u_k+\alpha_k}{N^u+K\alpha_k}$
  end for
end for
for each aspect k and item i **do**
  estimate $\beta_{k,i} = \frac{N^i_k+\eta_k}{N_k+I\eta_k}$
end for

5.3 Kullback-Leibler Divergence

The next two approximate inference algorithms we will consider both attempt to find an approximate posterior q that is as close as possible to the true posterior p. To do so they need a measure of divergence between distributions. Kullback-Leibler divergence is one such measure (Kullback, 1959). For two distributions q and p, the Kullback-Leibler (KL) divergence $\mathrm{KL}\left(p \,||\, q\right)$ is a measure of the information lost when q is used to approximate p. Originally given in 1951, the KL-divergence between two distributions is defined as (Kullback & Leibler, 1951)

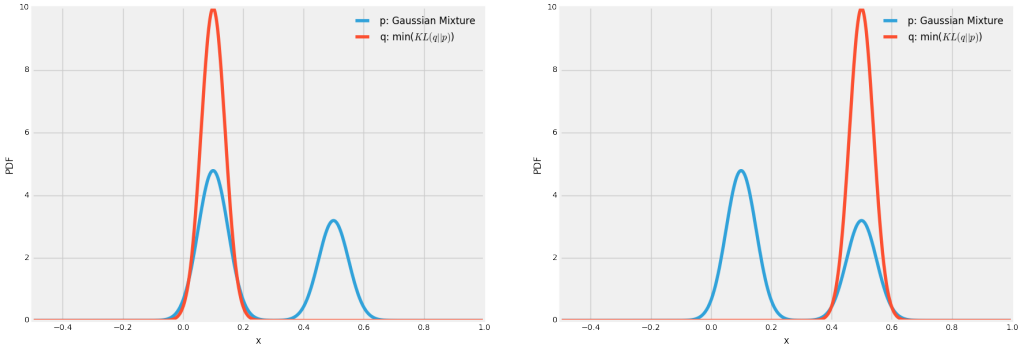
$$\begin{aligned}\mathrm{KL}\left(p \,||\, q\right) &= -\int \mathrm{p}(\boldsymbol{x})\ ln(q(\boldsymbol{x}))d\boldsymbol{x} + \left(\int \mathrm{p}(\boldsymbol{x})\ ln(\mathrm{p}(\boldsymbol{x}))d\boldsymbol{x}\right) \\ &= -\int \mathrm{p}(\boldsymbol{x}) ln\left\{\frac{q(\boldsymbol{x})}{\mathrm{p}(\boldsymbol{x})}\right\} d\boldsymbol{x}\end{aligned} \tag{5.23}$$

Kullback-Leibler divergence is non-negative $\mathrm{KL}(\cdot||\cdot) \geq 0$ and also non-symmetric $\mathrm{KL}\left(p \,||\, q\right) \neq \mathrm{KL}\left(q \,||\, p\right)$ (Shlens, 2014).

$$\mathrm{KL}\left(q \,||\, p\right) = -\int q(\boldsymbol{x}) ln\left\{\frac{\mathrm{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\right\} d\boldsymbol{x} \tag{5.24}$$

5.3.1 Minimising Kullback-Leibler Divergence

An important class of inference algorithms known as variational methods perform inference by finding approximate posteriors in cases where the true posterior is intractable to compute. These methods include Variational Inference and Expectation Propagation, and they find these approximations to the true posterior by minimising the KL divergence between the approximate q and true posterior p. However, because KL divergence is not symmetrical, the choice of using the KL or reversed KL divergence becomes important as it will influence the approximate posterior that we find.
To illustrate consider a Gaussian approximation q to a multimodal mixture of two Gaussians p, when we consider $\mathrm{KL}\left(q \,||\, p\right)$ figure 5.1 shows the two possible optimal solutions that minimise $\mathrm{KL}\left(q \,||\, p\right)$:



(a) (b)

Figure 5.1: A Gaussian approximation to a mixture of two Gaussian distributions so that the approximate distribution q minimises the Kullback-Leibler divergence $\mathrm{KL}\left(q \,||\, p\right)$. Note how q seeks out one of the modes of p and finds a good approximation in that region only.

The optimal solution seeks out one of the modes. This makes sense if we consider equation 5.24: the KL divergence is large if q is far from zero and p is close to it. This leads to approximations that avoid areas where p is small. Techniques that minimise $\mathrm{KL}\left(q \,||\, p\right)$ are *mode seeking* as they will find and approximate a posterior that seeks out one of the modes of the original posterior. Variational Inference is an approach to inference that seeks to find approximate distributions that minimise $\mathrm{KL}\left(q \,||\, p\right)$ (Blei *et al.*, 2017).

Considering the other case KL $(p \,||\, q)$, the approximation will avoid being small in areas where p has any mass leading to it averaging across the modes as seen in figure 5.2. These approaches are *moment matching* as they average over the modes of p in such a way that the moments of q match those of p. It is possible to define inference procedures that minimise KL $(p \,||\, q)$ and Expectation Propagation is one such approach (Bishop, 2006).

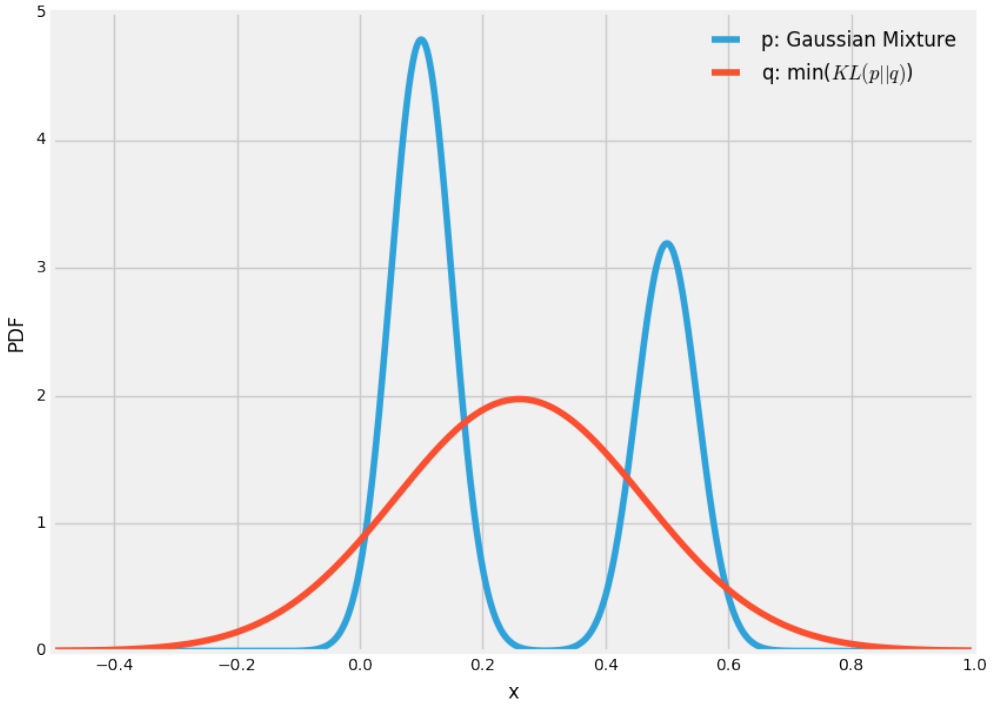


Figure 5.2: A Gaussian approximation to a mixture of two Gaussian distributions so that the approximate distribution q minimises the Kullback-Leibler divergence $KL(p||q)$. Note how the mean and variance of p and q are the same, despite the fact that q only has a single mode.

5.4 Variational Inference

Variational Inference (VI) re-frames inference into an optimization problem (Blei, 2011). The approach is based on the *calculus of variations*, which is

concerned with finding the derivatives of *functionals*. A functional can be thought of as a mapping that takes a function as the input and returns the value of the functional as output (Bishop, 2006). Many problems can be stated as an optimisation problem where the objective is to explore all possible input functions and find one that minimises or maximises a functional. These *variational methods* are not naturally approximate but become so because it is often necessary to restrict the possible input functions (Bishop, 2006). Variational inference uses these variational methods to perform approximate inference, where the goal is to find an approximation $q(\boldsymbol{\zeta})$ to the desired posterior (or any conditional) $\mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})$. For a comprehensive review of variational inference, see Blei *et al.* (2017).

Variational inference specifies a family of distributions $\mathcal{Q}$ over the latent variables $\boldsymbol{\zeta}$; each member of the family $q(\boldsymbol{\zeta}) \in \mathcal{Q}$ is a candidate approximation to the exact conditional (for inference we wish to approximate the posterior $\mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})$). The goal is to find the candidate that minimises the KL divergence to the posterior, thus transforming inference to the optimisation problem (Blei *et al.*, 2017; Fox & Roberts, 2012):

$$q^*(\boldsymbol{\zeta}) = \arg\min_{q(\boldsymbol{\zeta})\in\mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right). \tag{5.25}$$

The approximate distribution $q^*(\cdot)$ that is found is the best approximation to the posterior available in the given family $\mathcal{Q}$. The more expressive the family the better the approximation will be, but this may come at increased computational cost. Because the approximating family determines the complexity of the problem, there is a trade-off between selecting a family that is as expressive as possible while still keeping the problem computationally tractable.

5.4.1 Evidence Lower Bound (ELBO)

The objective in 5.25 is not tractable for the same reason the posterior is not tractable: it requires calculating the marginal likelihood. We then turn to an alternative objective that is equal to the KL divergence up to a constant. By defining a function that is equal to the negative KL divergence plus the log-evidence, we find a suitable objective. This objective is call the *evidence lower bound* (ELBO), and maximising the ELBO is equivalent to minimising the KL

divergence (Blei *et al.*, 2017). To get the ELBO, take the KL divergence

$$\mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right) = \mathbb{E}_q\left[\log q(\boldsymbol{\zeta})\right] - \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right] \tag{5.26}$$

and expand the conditional to show the dependence on $\mathrm{p}(\boldsymbol{x})$

$$\mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right) = \mathbb{E}_q\left[\log q(\boldsymbol{\zeta})\right] - \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{\zeta}, \boldsymbol{x})\right] + \log\, \mathrm{p}(\boldsymbol{x}). \tag{5.27}$$

The ELBO is the negative KL divergence plus the log evidence:

$$\begin{aligned} \mathrm{ELBO}(q) &= -\mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right) + \log\, \mathrm{p}(\boldsymbol{x}) \\ &= -\left(\mathbb{E}_q\left[\log q(\boldsymbol{\zeta})\right] - \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{\zeta}, \boldsymbol{x})\right]\right) + \log\, \mathrm{p}(\boldsymbol{x}) + \log\, \mathrm{p}(\boldsymbol{x}) \\ &= \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{\zeta}, \boldsymbol{x})\right] - \mathbb{E}_q\left[\log q(\boldsymbol{\zeta})\right] \end{aligned} \tag{5.28}$$

The ELBO gets its name because it lower-bounds the log evidence. By rewriting equation 5.28

$$\log\, \mathrm{p}(\boldsymbol{x}) = \mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})\right) + \mathrm{ELBO}(q) \tag{5.29}$$

and following from the fact that the KL divergence is always positive ($\mathrm{KL}(\cdot) \geq 0$), the ELBO must lower-bound the evidence:

$$\log\, \mathrm{p}(\boldsymbol{x}) \geq ELBO(q) \quad \forall q \in \mathcal{Q}. \tag{5.30}$$

5.4.2 ELBO as an Objective Function

By rewriting the ELBO as the sum of the expected log likelihood of the data and the KL divergence between the prior $\mathrm{p}(\boldsymbol{\zeta})$ the approximating distribution $q(\boldsymbol{\zeta})$, we can gain some insight into the optimal variational density (Blei *et al.*, 2017) and understand what parameters that maximising the ELBO will encourage putting mass on

$$\begin{aligned} ELBO(q) &= \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{\zeta})\right] + \mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\right] - \mathbb{E}_q\left[\log q(\boldsymbol{\zeta})\right] \\ &= \underbrace{\mathbb{E}_q\left[\log\, \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\right]}_{\text{(a)}} - \underbrace{\mathrm{KL}\left(q(\boldsymbol{\zeta}) \,||\, \mathrm{p}(\boldsymbol{\zeta})\right)}_{\text{(b)}}. \end{aligned} \tag{5.31}$$

The first term of the ELBO in equation 5.31:(a) is an expected likelihood and encourages approximations that place mass on configurations of latent variables that will give high probabilities to the observed data (Blei *et al.*, 2017).

The second term 5.31:(b) is the negative KL divergence between the approximation and the prior that will encourage approximations that are close to the prior (Blei *et al.*, 2017). The ELBO mirrors Bayes' rule and the balance between placing mass on parameters to increase the likelihood and being penalised for moving far from the prior without sufficient support.

5.4.3 Relationship to Expectation-Maximisation

Expectation-Maximisation (EM) is an algorithm designed for finding maximum likelihood estimates in models with latent variables (Blei *et al.*, 2017). The first term in the ELBO (equation 5.31(a)) is the expected log-likelihood: $\mathbb{E}_q\left[\log \mathrm{p}(\boldsymbol{x}|\boldsymbol{\zeta})\right]$, which is optimised by EM. EM alternates an E-Step which calculates the expected complete log-likelihood, by assuming that the expectations of the posterior $\mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})$ can be calculated, and an M-Step that optimizes it with respect to the model parameters.

VI differs from EM in that it does not assume that the expectations under $\mathrm{p}(\boldsymbol{z}|\boldsymbol{x})$ are computable and VI does not estimate fixed model parameters. VI then applies to models where the exact conditional of the latent variables cannot be calculated.

For cases where we wish to estimate fixed model parameters, such as for estimating prior parameters as with Empirical Bayes, we can use *variational EM*. Variational EM is an alternating procedure that maximises a lower bound with respect to the variational parameters and then maximises the lower bound with respect to the fixed parameters.

5.4.4 Mean Field Approximation: Factorized Distributions

The ELBO gives an objective for inference as optimisation, but first the variational family $\mathcal{Q}$ must be specified. The complexity of the chosen family dic-

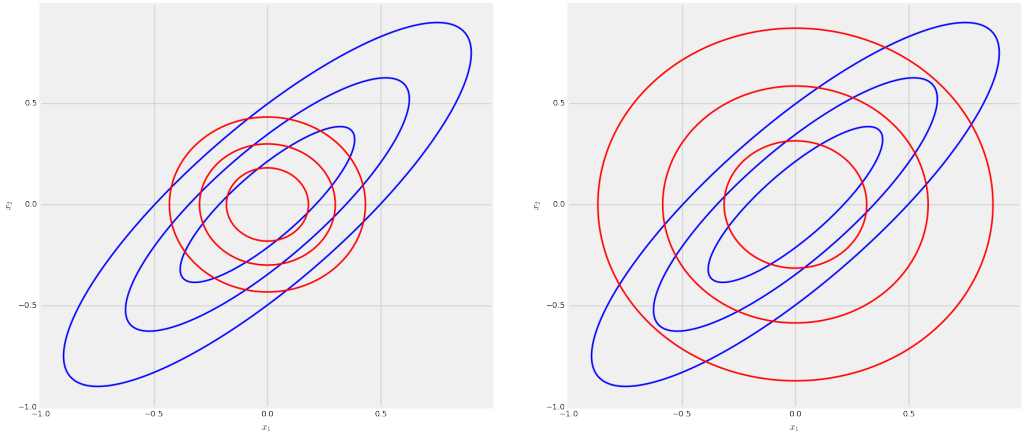
tates how complex the optimisation problem becomes (Blei, 2011). A common family and the family we will use is the *mean-field variational family* which assumes that the latent variables are mutually independent and each has a corresponding factor in the variational approximate density:

$$q(\boldsymbol{\zeta}) = \prod_j q_j(\zeta_j). \tag{5.32}$$

Each of the latent variables ζ_j is governed by a distinct variational factor $q_j(\zeta_j)$ (Blei *et al.*, 2017). During optimization these are the factors chosen so that the ELBO is maximised. The variational factors can have any parametric form that is appropriate for the random variable it is approximating. In many models the properties of the model will determine the optimal form of the variational factors (Blei *et al.*, 2017). For example, if ζ_j is a continuous variable, the factor $q_j(\zeta_j)$ might take the form of a Normal distribution with variational parameters μ_j and σ_j.

The power of the mean-field family is that it can capture the marginal distributions of the latent variables but it is limited in that it cannot capture correlation between them (Blei *et al.*, 2017). Figure 5.3 shows how the mean-field affects the approximation found for a bivariate Gaussian with correlated variables.

The optimal mean-field approximation to a bivariate Gaussian is the product of two Gaussians (Bishop, 2006). We can see that the approximation found by minimising $\mathrm{KL}\,(q \,||\, p)$ in figure 5.3(a) has the correct mean (it is mode seeking) but underestimates the variance significantly. This is understandable as the KL divergence will penalise the approximation for putting mass outside of the posterior, but this causes the approximate distribution to not have support over much of the posterior.



(a) (b)

Figure 5.3: A bivariate Gaussian p (blue) with a mean-field approximation q (red) that is found by minimising (a): $\mathrm{KL}(q \,||\, p)$. Note how the solutions are still mode-seeking and moment-matching even when p is not multimodal. (b): $\mathrm{KL}(p \,||\, q)$

We can also look at the mean-field approximation that minimises $\mathrm{KL}(p \,||\, q)$ as seen in figure 5.3(b). The approximate density now correctly matches the mean and variance (it is moment-matching) of the true distribution, but now places mass outside of the support of the posterior. There is therefore a trade-off where VI will have a good approximation at one of the modes of the posterior but not fully capture the posterior. In reverse, when using a moment-matching approximation, the approximate posterior covers the full support but also large areas where the posterior has no support. Which approximation is better depends on the model in question.

5.4.5 The Optimization Procedure: Co-ordinate Ascent Variational Inference (CAVI)

The final piece of Variational Inference is the optimisation algorithm. One of the most common algorithms is Coordinate ascent mean-field variational inference (CAVI) (Bishop, 2006). CAVI climbs the ELBO to a local optimum. CAVI updates each of the latent variables in turn, much like Gibbs sampling. Consider the j-th latent variable ζ_j; we can get its complete conditional given

all the other latent variables and the observations $\mathrm{p}(\zeta_j|\boldsymbol{\zeta}_{-j}, \boldsymbol{x})$. If the other variational factors $q_{-j}(\boldsymbol{\zeta}_{-j})$ are fixed, the optimal variational factor $q_j(z_j)$ is then proportional to (Blei *et al.*, 2017)

$$q_j^*(\boldsymbol{\zeta}_j) \propto \exp\left\{E_{q_{-j}}\left[\log \mathrm{p}(\zeta_j|\boldsymbol{\zeta}_{-j}, \boldsymbol{x})\right]\right\}. \tag{5.33}$$

This is a valid coordinate update because the expectations on the right hand side do not involve the $j-th$ variational factor. The factors are independent as assumed by the mean-field family.

The algorithm for CAVI is given in algorithm 6 (Blei *et al.*, 2017).

Algorithm 6 Coordinate Ascent Mean Field Variation Inference

Input: A model $\mathrm{p}(\boldsymbol{x}, \boldsymbol{z})$ and observed data $\boldsymbol{x}$
Output: A variational approximation distribution $q(\boldsymbol{z}) = \prod_j q_j(z_j)$
Initialise variational factors
while ELBO not converged **do**
  for $q_j(\zeta_j) \in q(\boldsymbol{\zeta})$ **do**
    $q_j(\zeta_j) \propto \exp\left\{E_{q_{-j}}\left[\log \mathrm{p}(\zeta_j|\boldsymbol{\zeta}_{-j}, \boldsymbol{x})\right]\right\}$
  end for
  Calculate $\mathrm{ELBO}(q) = E_q\left[\log \mathrm{p}(\boldsymbol{\zeta}, \boldsymbol{x})\right] + E\left[\log q(\boldsymbol{\zeta})\right]$
end while

5.4.6 Relationship with Gibbs Sampling

The CAVI updates are closely related to Gibbs sampling. A Gibbs sampler keeps a value for each latent variable and samples from each variable's complete conditional. CAVI uses the same complete conditional but takes the expected log to iteratively set each latent variable's variational factor (Blei *et al.*, 2017):

$$q(\zeta_j) \propto \exp\left\{\mathbb{E}\log\left(\mathrm{p}(\zeta_j|\zeta_{-j})\right)\right\}. \tag{5.34}$$

Much like Gibbs sampling in VI, variables can also then be collapsed out; Teh *et al.* (2007) show a version of collapsed VI for Latent Dirichlet Allocation.

5.4.7 Variational Inference with the Exponential Family

If each of the complete conditionals is in the Exponential family, VI is greatly simplified and CAVI is easier to derive (Blei *et al.*, 2017). Each complete conditional is then given by a distribution in the canonical form of the exponential family (equation 2.32):

$$\mathrm{p}(\zeta_j|\boldsymbol{\zeta}_{-j}, \boldsymbol{x}) = h(\zeta_j)\exp\left\{\eta_j(\boldsymbol{\zeta}_{-j}, \boldsymbol{x})^T\zeta_j - a(\eta_j(\boldsymbol{\zeta}_{-j}, \boldsymbol{x}))\right\}. \tag{5.35}$$

Where ζ_j is its own sufficient statistic, $h(\cdot)$ is a base measure, and $a(\cdot)$ the log normaliser; because it is a conditional, $\eta_j(\boldsymbol{z}_{-j}, \boldsymbol{x})$ is a function of $(\boldsymbol{z}_{-j}, \boldsymbol{x})$ (Blei *et al.*, 2017). Mean field VI is then simple: consider the CAVI update from equation 5.33 with the conditional in the exponential family (Blei, 2011)

$$\begin{aligned} q(\zeta_j) &\propto \exp\left\{\mathbb{E}_q\left[\log \mathrm{p}(\zeta_j|\boldsymbol{\zeta}_{-j}, \boldsymbol{x})\right]\right\} \\ &= \exp\left\{\log h(\zeta_j) + \mathbb{E}_q\left[\eta_j(\boldsymbol{\zeta}_{-j}, \boldsymbol{x})\right]^T\zeta_j - \mathbb{E}\left[a(\eta_j(\boldsymbol{\zeta}_{-j}, \boldsymbol{x}))\right]\right\} \\ &\propto h(\zeta_j)\exp\left\{\mathbb{E}\left[\eta_j(\boldsymbol{\zeta}_{-j}, \boldsymbol{x})\right]^T\zeta_j\right\} \end{aligned} \tag{5.36}$$

The update shows that the optimal parametric form of the variational factors is in the same exponential family as the corresponding complete conditional - its parameter will have the same dimension, it has the same base measure and log normaliser (Blei *et al.*, 2017). The CAVI update for the j-th variational parameter v_j is equal to the expectation of the natural parameter for ζ_j

$$v_j = \mathbb{E}\left[\eta(z_j, \boldsymbol{x})\right]. \tag{5.37}$$

5.4.8 Variational Inference for Latent Dirichlet Allocation

Blei *et al.* (2003) derived a variational procedure for Latent Dirichlet Allocation, which we have summarised here. For full derivations, see (Blei *et al.*, 2003). LDA is a conditionally conjugate model that has local and global variables. The local variables are per-observation latent variables and the global variables act as parameters. CAVI for conditionally conjugate models alternates between updating the local variational parameters and updating the

global ones (Blei *et al.*, 2017).

We place variational distributions on the parameters $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and $\boldsymbol{z}$, so that the variational approximation to each aspect item distribution $\boldsymbol{\beta}_k$ is a Dirichlet with a variational parameter λ_k:

$$q(\boldsymbol{\beta}_k) = \text{Dir}(\boldsymbol{\beta}_k|\boldsymbol{\lambda}_k). \tag{5.38}$$

Similarly $q(\boldsymbol{\theta}_u)$ is given as a Dirichlet with parameter $\boldsymbol{\gamma}_u$:

$$q(\boldsymbol{\theta}_u) = \text{Dir}(\boldsymbol{\theta}_u|\boldsymbol{\gamma}_u). \tag{5.39}$$

Finally the variational distribution over each of the N aspect assignments for a user u, $q(z_{u,n})$, has a multinomial parameter $\boldsymbol{\phi}_{u,n}$:

$$q(z_{u,n}) = \text{Multi}(\boldsymbol{\phi}_{u,n}). \tag{5.40}$$

The variational approximation to the LDA posterior is then:

$$q(\boldsymbol{\beta}, \boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{\lambda}, \boldsymbol{\phi}, \boldsymbol{\gamma}) = \prod_{k}^{K} \text{Dir}(\boldsymbol{\beta}_k|\boldsymbol{\lambda}_k) \prod_{u}^{U} q_u(\boldsymbol{\theta}_u, \boldsymbol{z}_u|\boldsymbol{\phi}_u, \boldsymbol{\gamma}_u). \tag{5.41}$$

The distribution $q_u(\boldsymbol{\theta}_u, \boldsymbol{z}_u|\boldsymbol{\phi}_u, \boldsymbol{\gamma}_u)$ can be expanded to get a user level variational distribution:

$$q_u(\boldsymbol{\theta}_u, \boldsymbol{z}_u|\boldsymbol{\phi}_u, \boldsymbol{\gamma}_u) = q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \prod_{n}^{N} q(z_n|\boldsymbol{\phi}_{u,n}). \tag{5.42}$$

We can start by first updating these local user-level parameters. First is the multinomial parameter $\phi_{u,n,k}$, which is the probability that the n-th item for the user was generated by aspect k. The update for $\phi_{u,n,k}$ is

$$\phi_{u,n,k} \propto \beta_{k,i_n} \exp\left\{E_q\left[\log(\theta_k)|\gamma\right]\right\}. \tag{5.43}$$

$\phi_{u,n}$ must sum to 1 to be a valid multinomial, thus we must normalise ϕ. The second user level parameter is the Dirichlet parameter $\boldsymbol{\gamma}_u$. The update equation of the k-th component of the posterior Dirichlet parameter γ_k is

$$\gamma_k = \alpha_k + \sum_{n}^{N} \phi_{u,n,k}. \tag{5.44}$$

The variational procedure for each user u is given in algorithm 7:

Algorithm 7 Variational parameter update procedure for user level parameters when performing inference for Latent Dirichlet Allocation

initialise $\phi_{u,n,k} = \frac{1}{K}$ for all n and k
initialise $\gamma_k = \alpha_k + N/k$ for all K aspects
repeat
  for n = 1 to N **do**
    for each aspect k **do**
      $\phi_{u,n,k}^{(t+1)} = \beta_{k,i_n} \exp\{E_q[\log(\theta_k)|\gamma]\}$
    end for
    normalise $\boldsymbol{\phi}_{u,n}$
  end for
  $\boldsymbol{\gamma}^{(t+1)} = \boldsymbol{\alpha} + \sum_n^N \phi_{u,n}^{(t+1)}$
until convergence

After updating each local variational parameter for each user, CAVI alternates to updating the global parameters. In LDA the global variables are the aspect distributions in $\boldsymbol{\beta}$. We placed the global aspect distribution variational parameter $\boldsymbol{\lambda}$ on $\boldsymbol{\beta}$, which we now update. The variational update for the item i in the k-th aspect is where $y_{u,i}$ is the implicit feedback count the user gave the item.

$$\lambda_{k,i} = \eta_k + \sum_{u}^{U} \sum_{n}^{N_u} \boldsymbol{\phi}_{u,n,i} y_{u,i} \tag{5.45}$$

Putting together the local and global updates to get a variational EM-like procedure as shown in algorithm 8.

Algorithm 8 Full Variational EM procedure for Latent Dirichlet Allocation

while ELBO not converged **do**

Local Variational Updates
for u=1 to U **do**
initialise $\phi_{u,n,k} = \frac{1}{K}$ for all u, n and k
initialise $\gamma_{u,k} = \alpha_k + N/k$ for all k
repeat
for each user u **do**
for n = 1 to N **do**
for i=1 to k **do**
$\phi_{u,n,k}^{(t+1)} = \beta_{k,i_n} \exp\{E_q[\log(\theta_k)|\gamma]\}$
end for
normalise $\boldsymbol{\phi}_{u,n}$
end for
end for
$\gamma_u^{(t+1)} = \boldsymbol{\alpha} + \sum_n^N \phi_{u,n}^{(t+1)}$
until convergence
end for

Global Variational Updates
for each aspect k **do**
$\lambda_{k,i} = \eta_k + \sum_u^U \sum_n^{N_u} \boldsymbol{\phi}_{u,n,i} y_{u,i}$
end for

end while

5.4.9 Stochastic Inference (Online Learning)

Coordinate ascent requires iterating through all the data for each iteration. For conditionally conjugate models like LDA, this is expensive and does not scale well. An alternative is to use *stochastic variational inference* (SVI), which uses gradient-based optimisation and follows the natural gradient of the ELBO. The stochastic approach allows us to cheaply compute a noisy approximation to the gradient and follow it to the same local optima as CAVI. Using SVI then allows us to create online or parallel versions of inference that scale far better than CAVI. CAVI is a batch algorithm that assesses each data point for each inference iteration; SVI is online and can perform an iteration with a single data point, allowing it to quickly converge after just assessing each data point once. The greatest advantage to SVI is that the update equations are

the same as CAVI, meaning that any CAVI algorithm can be easily converted to an online SVI one (Blei *et al.*, 2017).

Hoffman *et al.* (2010) shows an online SVI algorithm for LDA. It can scale to millions of documents and is as accurate as LDA models found with other inference algorithms.

5.4.10 Automatic Differentiation Variational Inference

Automatic Differentiation Variational Inference (ADVI) automatically derives a VI algorithm using only a model definition and a dataset (Kucukelbir *et al.*, 2016). This is enormously helpful as new models can be tried out without having to derive inference algorithms for them. Models defined in the probabilistic programming language Stan can have inference derived automatically with ADVI (Kucukelbir *et al.*, 2015).

5.5 Expectation Propagation (EP)

Originally given by (Opper & Winther, 2000) and generalized by (Minka, 2001), Expectation Propagation (EP) is a variant of message passing algorithms for inference. Message passing algorithms infer the target density. In our case we are interested in the posterior by using a collection of localized inferences (Gelman *et al.*, 2014).

EP builds on a previous form of approximate inference called Assumed Density Filtering (ADF). ADF approximates the posterior with an approximate distribution $q(\boldsymbol{\zeta})$. ADF starts $q(\boldsymbol{\zeta})$ at the prior on $\boldsymbol{\zeta}$ and iterates through each data point, incorporating the point into the approximate posterior. EP extends the ADF procedure to allow multiple passes through the data (Blei, 2005).

Much like Variational Inference EP finds an approximate density from some specific parametric family $q(\boldsymbol{\zeta})$ that approximates the true density $\mathrm{p}(\boldsymbol{\zeta})$. The approximation is found by minimizing the Kullback-Leibler divergence $\mathrm{KL}\left(p \,||\, q\right)$ (Bishop, 2006); this is the reverse KL compared to VI. See figure 5.2 for an

example of minimising KL $(p \,||\, q)$ - we expect EP to average across the modes of the posterior.

When performing inference we are interested in the joint distribution $\mathrm{p}(\boldsymbol{x}, \boldsymbol{\zeta})$ and the posterior $\mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x})$. Much like VI, EP assumes that the target distribution (here the joint distribution) has a convenient factorization as an unnormalised product of terms

$$\mathrm{p}(\boldsymbol{x}, \boldsymbol{\zeta}) \propto \prod_i t_i(\boldsymbol{\zeta}). \tag{5.46}$$

For inference, since we are interested in the posterior, EP assumes the posterior form (Bishop, 2006)

$$\begin{aligned} \mathrm{p}(\boldsymbol{\zeta}|\boldsymbol{x}) &= \frac{1}{\mathrm{p}(\boldsymbol{x})} \prod_i t_i(\boldsymbol{\zeta}) \\ &= \frac{1}{\int \prod_i t_i(\boldsymbol{\zeta}) d\boldsymbol{\zeta}} \prod_i t_i(\boldsymbol{\zeta}). \end{aligned} \tag{5.47}$$

This brings to mind the mean-field approximation from variational inference. However, EP is assuming that the target distribution also factorises. The EP approximation to the posterior $q(\boldsymbol{\zeta})$ also forms a product of factors:

$$q(\boldsymbol{\zeta}) = \frac{1}{Z} \prod_i \tilde{t}_i(\boldsymbol{\zeta}). \tag{5.48}$$

where each $\tilde{t}_i(\boldsymbol{\zeta})$ is an approximation corresponding to a factor term in the true posterior and Z is a normalizer.

EP performs inference by iteratively optimizing each factor in the context of all other factors. EP initializes each factor $\tilde{t}_i(\boldsymbol{\zeta})$, then iterates through each factor in turn, refining the approximation in a similar fashion to the variational update (Bishop, 2006). To refine the approximation to a single factor $\tilde{t}_i(\boldsymbol{\zeta})$, we remove its contribution from the approximate distribution to get an unnormalised *cavity distribution*:

$$q_{\setminus i}(\boldsymbol{\zeta}) \propto \frac{q(\boldsymbol{\zeta})}{\tilde{t}_i(\boldsymbol{\zeta})}.. \tag{5.49}$$

Calculate the *titled distribution* defined as:

$$q_{-i} \propto t_i(\boldsymbol{\zeta}) q_{\setminus i}(\boldsymbol{\zeta}). \tag{5.50}$$

EP then constructs a new approximation $q^{new}(\boldsymbol{\zeta})$ by matching moments to the titled distribution.

Then finds an updated approximation to the target factor $t_i(\boldsymbol{\zeta})$ as

$$\tilde{t}_i^{new}(\boldsymbol{\zeta}) = Z_i \frac{q^{new}(\boldsymbol{\zeta})}{q_{\setminus i}(\boldsymbol{\zeta})}. \tag{5.51}$$

The normalising factor can be calculated as:

$$Z_i = \int q_{-i}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}. \tag{5.52}$$

EP can be given in general as in algorithm 9 (Gelman *et al.*, 2014).

Algorithm 9 General Expectation Propagation

initialize the term factor approximation $\tilde{t}_i(\boldsymbol{\zeta})$
repeat
  for $k = 1 to K$ **do**
    Calculate the cavity distribution $q_{\setminus i}(\boldsymbol{\zeta}) = \frac{q(\boldsymbol{\zeta})}{\tilde{t}_i(\boldsymbol{\zeta})}$
    Calculate the titled distribution $q_{-i} = t_i(\boldsymbol{\zeta}) q_{\setminus i}(\boldsymbol{\zeta})$
    Calculate $q^{new}(\boldsymbol{\zeta})$ by matching moments to the q_{-i}
    Calculate the normaliser $Z_i = \int q_{-i}(\boldsymbol{\zeta}) d\boldsymbol{\zeta}$
    Get the new term approximation $\tilde{t}_i^{new}(\boldsymbol{\zeta}) = Z_i \frac{q^{new}(\boldsymbol{\zeta})}{q_{\setminus i}(\boldsymbol{\zeta})}$
  end for
until all factor approximations have converged

By matching the moments $\tilde{t}_i(\boldsymbol{\zeta})$ is chosen so that the KL divergence between $\tilde{t}_i(\boldsymbol{\zeta}) q_{\setminus i}(\boldsymbol{\zeta})$ and $t_i(\boldsymbol{\zeta}) q_{\setminus i}(\boldsymbol{\zeta})$ is minimized (Minka & Lafferty, 2002).

$$\tilde{t}_i^{new}(\boldsymbol{\zeta}) = \arg\min \mathrm{KL}\left(t_i(\boldsymbol{\zeta}) q_{\setminus i}(\boldsymbol{\zeta}) \,||\, \tilde{t}_i(\boldsymbol{\zeta}) q_{\setminus k}(\boldsymbol{\zeta})\right) \tag{5.53}$$

which corresponds to minimizing $\mathrm{KL}\left(q_{\setminus}(\boldsymbol{\zeta}) \,||\, q(\boldsymbol{\zeta})\right)$.

5.5.1 Minimizing KL Divergence

To understand how matching the moments minimises the KL divergence, we can consider the case where the factor terms are constrained to belong to the Exponential family. Constraining the terms also ensures that computing EP remains tractable. If we consider the problem of minimizing $\mathrm{KL}(p \,||\, q)$ where

q belongs to the exponential family and is over $\boldsymbol{\zeta}$, any distribution in the Exponential family can be written in the form:

$$q(\boldsymbol{\zeta}) = h(\boldsymbol{\zeta})g(\boldsymbol{\eta})\exp\left\{\boldsymbol{\eta}^T\boldsymbol{u}(\boldsymbol{\zeta})\right\} \tag{5.54}$$

the KL divergence then becomes the following as a function of $\boldsymbol{\eta}$ (Bishop, 2006)

$$\mathrm{KL}(p\,||\,q) = \ln g(\boldsymbol{\eta}) - \boldsymbol{\eta}^T\mathbb{E}_p[\boldsymbol{u}(\boldsymbol{\zeta})] + \text{const.} \tag{5.55}$$

To minimize $\mathrm{KL}(p\,||\,q)$ we set the gradient of 5.55 with respect to $\boldsymbol{\eta}$ as zero, to get

$$-\bigtriangledown \ln g(\boldsymbol{\eta}) = \mathbb{E}_p[\boldsymbol{u}(\boldsymbol{\zeta})] \tag{5.56}$$

but the negative gradient of $\ln g(\boldsymbol{\eta})$ is equal to expectation of $g(\boldsymbol{\eta})$ under q (Bishop, 2006). The optimal solution is then given by matching the moments (sufficient statistics) of p and q.

$$\mathbb{E}_{q(\boldsymbol{\zeta})}\left[\boldsymbol{u}(\boldsymbol{\zeta})\right] = \mathbb{E}_{\mathrm{p}(\boldsymbol{\zeta})}\left[\boldsymbol{u}(\boldsymbol{\zeta})\right]. \tag{5.57}$$

This is a cheap operation compared to directly minimizing the KL divergence. It also matches our intuition when we consider a factorised approximation minimises $KL(p||q)$ as in figure 5.3(b). The factorised approximation successfully matches the moments of the target distribution at the cost having having support where the target does not.

5.5.2 Convergence

EP minimizes the local KL divergences on the factors, not the global objective, unlike VI. Therefore it is not guaranteed to converge. Techniques have been developed that directly optimize the objective (Opper & Winther, 2005), but they are significantly slower than EP (Bishop, 2006).

5.5.3 Relationship with Loopy Belief Propagation

Minka *et al.* (2005) shows that many message-passing algorithms can be derived by minimizing divergences from the alpha family (Bishop, 2006). When operating on graphs, EP can be used to approximate a belief network with a simpler network with fewer edges, and when the approximation is completely

disconnected, EP is equivalent to loopy belief propagation (LBP) (Minka, 2001).

5.5.4 Expectation Propagation for Latent Dirichlet Allocation

Minka & Lafferty (2002) gives an example of EP for the LDA model. EP is used to estimate the posterior distribution on the aspect mixture $\boldsymbol{\theta}_u$

$$\mathrm{p}(\boldsymbol{\theta}_u|\boldsymbol{i}_u) = \frac{\mathrm{p}(\boldsymbol{\theta}_u)\prod_{i\in\boldsymbol{i}_u}\sum_{k=1}^{K}\beta_{k,i}\theta_{u,k}}{\int \mathrm{p}(\boldsymbol{\theta}_u)\prod_{i\in\boldsymbol{i}_u}\sum_{k=1}^{K}\beta_{k,i}\theta_{u,k}d\boldsymbol{\theta}_u} \tag{5.58}$$

the approximate posterior is given as a single Dirichlet on $\boldsymbol{\theta}_u$ with parameter $\boldsymbol{\gamma}_u$

$$q(\boldsymbol{\theta}_u) = \frac{\Gamma(\sum_{k=1}^{K}\gamma_{u,k})}{\prod_{k=1}^{K}\Gamma(\gamma_{u,k})}\prod_{k=1}^{K}\theta_{u,k}^{\gamma_{u,k}-1} \tag{5.59}$$

write the factorised form of q which takes the form:

$$q(\boldsymbol{\theta}_u) \propto \mathrm{p}(\boldsymbol{\theta}_u)\prod_{i\in\boldsymbol{i}_u}\tilde{t}_i(\boldsymbol{\theta}_u). \tag{5.60}$$

Where it has been factorised into terms in the form:

$$\tilde{t}(\boldsymbol{\theta}_u) = s_i\prod_{k=1}^{K}\theta_{u,k}^{\boldsymbol{\beta}_{k,i}}. \tag{5.61}$$

$\boldsymbol{\gamma}_u$ can be computed given $bm\beta$ (Blei, 2005)

$$\gamma_{u,k} = \alpha_k + \sum_{i=1}^{I}\beta_{k,i}. \tag{5.62}$$

To apply EP, choose an item i_n for the user with which we wish to update the posterior

1. Calculate the cavity distribution without the contribution of the item $i_{u,n}$, as a Dirichlet with parameter $\gamma_{\setminus n}$

$$q_{\setminus n}(\boldsymbol{\theta}_u) \propto \mathrm{p}(\boldsymbol{\theta}_u)\prod_{i\neq n}\tilde{t}_i(\boldsymbol{\theta}_u). \tag{5.63}$$

2. Calculate the approximate posterior

$$\tilde{p}(\boldsymbol{\theta}_u|\boldsymbol{y}_u) \propto \sum_{k=1}^{K} \boldsymbol{\beta}_{k,i}\theta_k q_{\backslash n}(\boldsymbol{\theta}_u) \tag{5.64}$$

3. Calculate the normalizer

$$\begin{aligned} Z_n &= \int \sum_{k=1}^{K} \mathrm{p}(i_n|k)\theta_k q_{\backslash n}(\boldsymbol{\theta}_u) d\boldsymbol{\theta}_u \\ &= \sum_{k=1}^{K} \mathrm{p}(i_n|k) \int \theta_k q_{\backslash n}(\boldsymbol{\theta}_u) d\boldsymbol{\theta}_u \\ &= \sum_{k=1}^{K} \mathrm{p}(i_n|k) \frac{(\boldsymbol{\gamma}_{\backslash n})_k}{\sum \boldsymbol{\gamma}_{\backslash n}} \end{aligned} \tag{5.65}$$

4. Minimize the KL divergence between $\tilde{p}$ and q by matching moments to get $\boldsymbol{\gamma}^{new}$. To do so, match the mean and variance of the Dirichlets with the following update:

$$m_k = \frac{1}{Z_n} \frac{(\boldsymbol{\gamma}_{\backslash n})_z}{\sum \boldsymbol{\gamma}_{\backslash n}} \frac{\mathrm{p}(i_n|k) + \sum_{k=1}^{K} \mathrm{p}(i_n|k)(\boldsymbol{\gamma}_{\backslash n})_k}{1 + \sum \boldsymbol{\gamma}_{\backslash n}} \tag{5.66}$$

$$m_k^{(2)} = \frac{1}{Z_n} \frac{(\boldsymbol{\gamma}_{\backslash n})_k}{\sum \boldsymbol{\gamma}_{\backslash n}} \frac{(\boldsymbol{\gamma}_{\backslash n})_k + 1}{\sum \gamma_{\backslash n} + 1} \frac{2\,\mathrm{p}(i_n|k) + \sum_{k=1}^{K} \mathrm{p}(i_n|k)(\boldsymbol{\gamma}_{\backslash n})_k}{2 + \sum \boldsymbol{\gamma}_{\backslash n}} \tag{5.67}$$

$$\gamma_k^{new} = m_z \left(\frac{\sum_{k=1}^{K} (m_k - m_k^{(2)})}{\sum_{k=1}^{K} (m_k^{(2)} - m_k^2)} \right) \tag{5.68}$$

5. Calculate the n-th approximating term $\tilde{t}_n$. Update $\boldsymbol{\beta}$ with step size $\lambda = \frac{1}{y_n}$

$$\beta_{k,i} = \lambda(\gamma_k^{new} - \boldsymbol{\gamma}_{-n}) + (1 - \lambda)\beta_{k,i}^{old} \tag{5.69}$$

$$s_i = Z_n \frac{\Gamma(\sum \gamma^{new}) \prod \Gamma(\gamma_{\backslash n})}{\prod \Gamma(\gamma_z^{new}) \Gamma(\sum \gamma_{\backslash n})} \tag{5.70}$$

6. Incorporate $\tilde{t}_n$ back into $q(\boldsymbol{\theta}_u)$ by scaling the change in $\boldsymbol{\beta}$

$$\gamma_k = \gamma_k^{old} + y_n(\beta_{k,i} - \beta_{k,i}^{old}) \tag{5.71}$$

Minka & Lafferty (2002) treat the aspects $\boldsymbol{\beta}$ as a fixed parameter and derive a variational EM algorithm to estimate them. The algorithm uses the approximate posterior on the user mixtures for the variational E step. The general process is:

E-Step Compute the approximate posterior for each document $q(\boldsymbol{\theta}_u)$

M-Step Maximise a lower bound with respect the log-likelihood to get a maximum likelihood estimate of $\boldsymbol{\beta}$

5.6 Verifying Approximate Inference

One challenge with unsupervised models like LDA is that the true distribution is not known for real-world datasets. This makes it challenging to verify whether an implemented inference algorithm is functioning correctly. To solve this we can check results of inference on artificial datasets with known parameters. One such artificial dataset was generated with 10 aspects over 1050 items.

Each aspect is shaped like a normal distribution with a standard deviation of 25 so that they each cover roughly 100 of the items. Each aspect is placed so that their modes are 100 items apart such that they overlap slightly, the last 50 items are evenly distributed between all the aspects as noise.

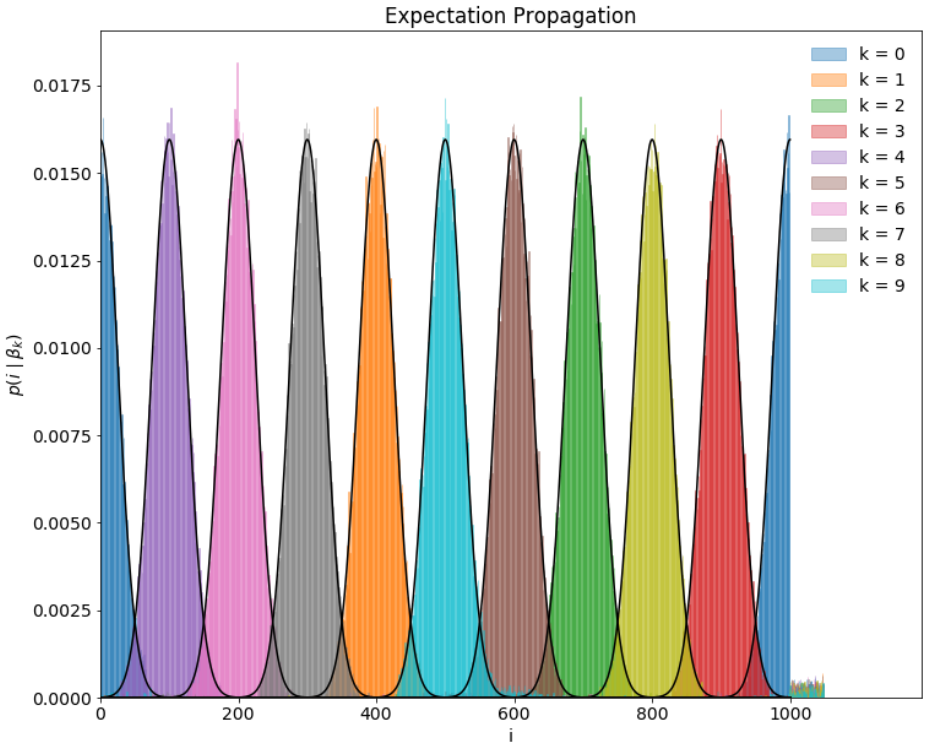


Figure 5.4: Aspect distributions from LDA inference with EP on an artificial dataset

10000 artificial users were generated, where each user had 30 items. VI and EP were used with uniform priors for the user and aspect priors. The inferred aspects can be seen in Figure 5.4 for EP and 5.5 for VI. With the large number of artificial users this should not be a challenging problem for the inference algorithms, if they are working correctly, and we can see that the inferred distributions closely match the ground truth distributions (shown as black curves).

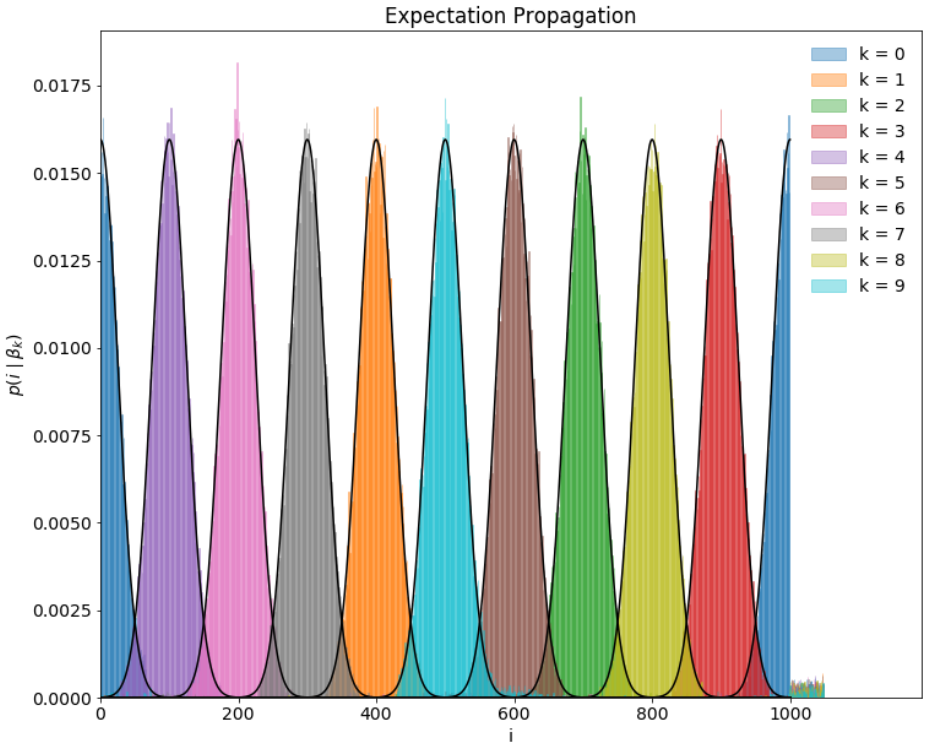


Figure 5.5: Aspect distributions from LDA inference with VI on an artificial dataset

5.6.1 Restricting the number of aspects

It is also interesting to explore the differences between the two approaches when we restrict the number of aspects. The true distribution is known to have ten aspects, but by restricting the LDA model to only four aspects we can gain some insight into the inner workings of the two approaches: EP and VI. Figure 5.6 and 5.7 show the the aspect item distribution parameters for the models learned with VI and EP respectively.

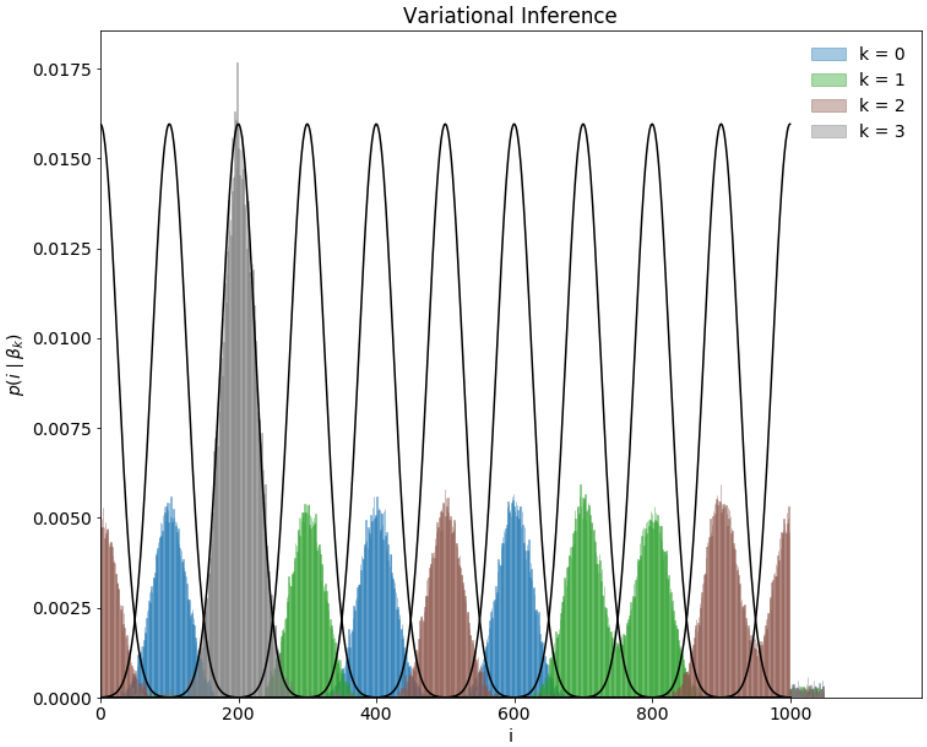


Figure 5.6: Aspect distributions from LDA inference with VI on an artificial dataset. The LDA was trained with 4 aspects when the artificial dataset is known to have 10.

It is interesting to note that EP prefers to spread the limited aspects over the true aspects whereas VI still has an aspect that is for a single aspect from the ground truth distribution.

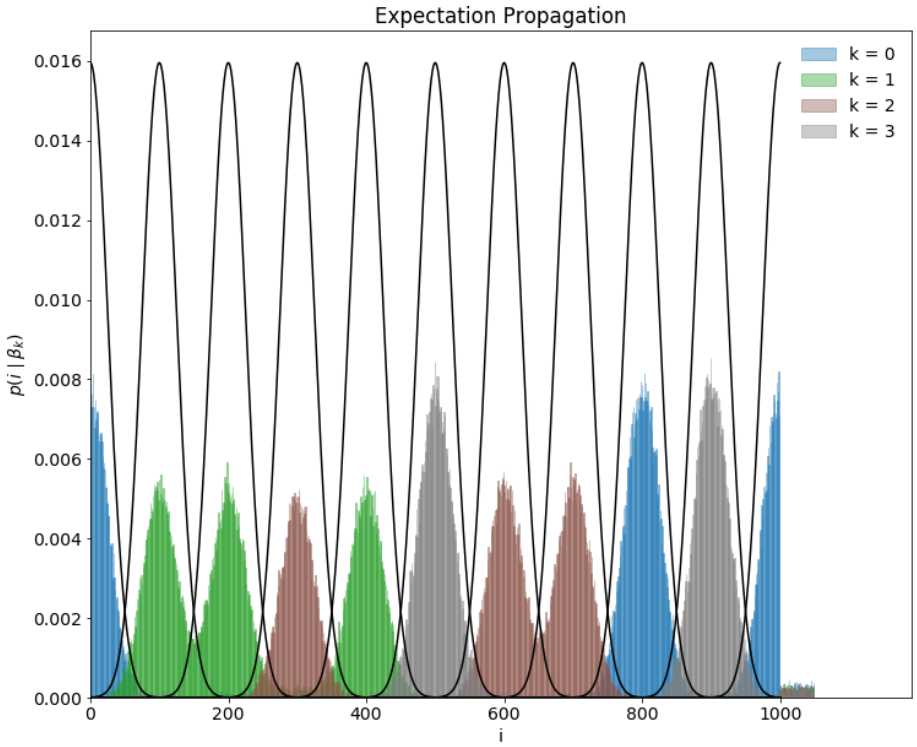


Figure 5.7: Aspect distributions from LDA inference with EP on an artificial dataset. The LDA was trained with 4 aspects when the artificial dataset is known to have 10.

5.6.2 Sparse user mixture prior

One feature of the artifical dataset is that is was set up so that every artificial user exhibited multiple aspects. If set the user mixture prior so that it is sparse ($\alpha_k = 0.0001$ as opposed to $\alpha_k = 0.1$ for a uniform prior) this will encourage the inference to spread the inferred aspects over multiple true aspects. This is because if the algorithm is encouraged to assume that a user exhibits just a single aspect, one would assume that the inferred aspect item distributions would be spread out to compensate.

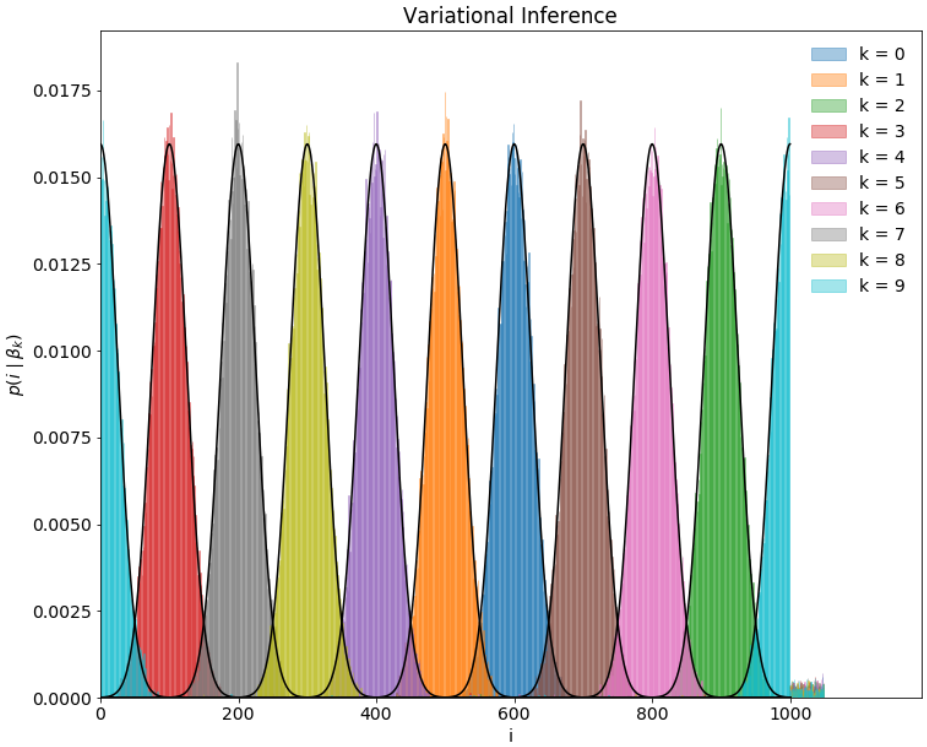


Figure 5.8: Aspect distributions from LDA inference with VI on an artificial dataset. The LDA was trained with 10 aspects when the artificial dataset is known to have 10. However the user mixture prior was set sparse when the true distribution is dense.

In Figure 5.8 and 5.8 we can see the inferred aspects. Despite the incorrect prior, VI still favours concentrating mass under the modes of the ground truth distributions. EP is more inclined to spread its mass out over the aspects. This is what one may expect.

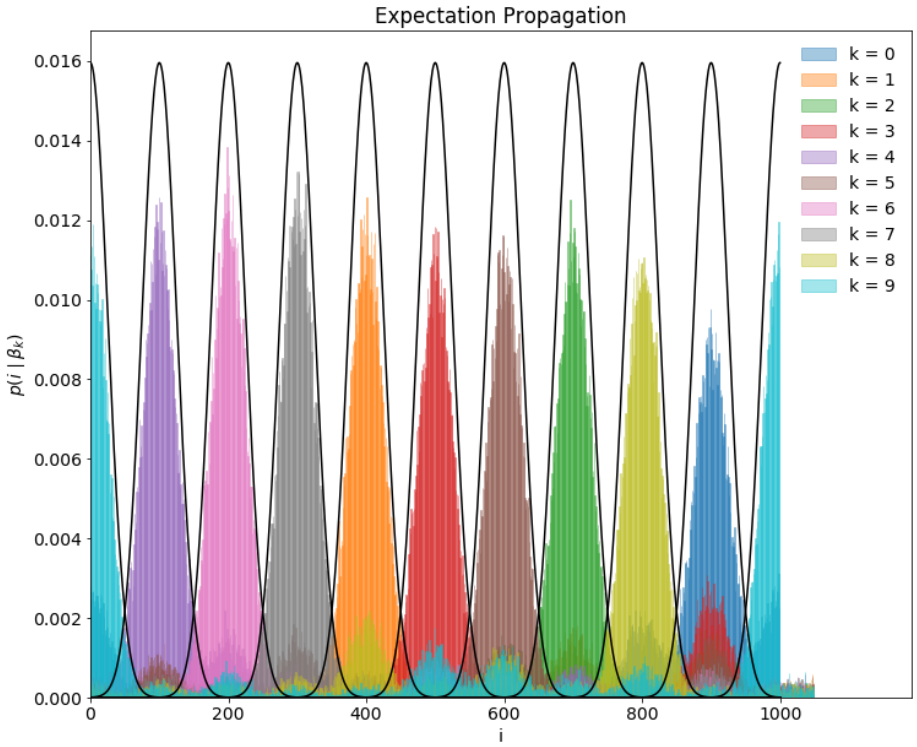


Figure 5.9: Aspect distributions from LDA inference with EP on an artificial dataset. The LDA was trained with 4 aspects when the artificial dataset is known to have 10. However the user mixture prior was set sparse when the true distribution is dense.

5.6.3 Further validation

Another check when working with LDA is to manually inspect the aspects and aspect mixtures inferred from known datasets. The Associated Press dataset is well studied in topic modelling and is known to produce easily interpretable topics with LDA. If the inferred topics are nonsensical or seemingly random there may be an error with the inference implementation.

Another strategy is inspecting whether known similar items are place closely together in aspect space. A good check here is to use sequels, for example in Figure 4.3 the *Star Wars* movies are placed closely together in aspect 15. This is a good sign as we expect these movies to frequently occur together.

However this approach is more of a sanity check and may not always be useful as a possible advantage of the model is finding non-obvious patterns that one would not have thought of.

When evaluating the traditional recommender approaches one can compare their RMSE scores for rating prediction on known datasets. As well as comparing them to popular open source implementations, such as in the *scikit-learn* Python package. For this project the matrix factorisation used the SVD modules from sci-kit as it is a known good implementation of SVD.

5.7 Discussion

The three inference algorithms we have seen are all based on very different fundamental grounds, but find surprisingly similar strategies. All are approaches to approximate inference, but the approximation for Gibbs sampling comes from limited computing power - we cannot run the chain for an infinite time to get true samples. Variational Bayes and Expectation Propagation both limit the family and form of the approximate posterior they find. Despite their difference the update equations often look similar between the different approaches, and they often are related to other inference techniques not discussed here. For example, Variational Inference is closely related to Expectation Maximization, and Expectation Propagation is closely related to graph-based inference techniques such as Loopy Belief Propagation.

An interesting difference is the choice of KL objective for VI and EP. The choice of KL will affect whether the approximate posterior is mode seeking or moment matching, but EP optimises the KL on the factors of the posterior whereas VI follows the objective over the whole posterior.

Chapter 6

Experimental Results

The goal of these experiments was to determine if LDA is suitable for recommendation when evaluated on top-N performance. The first set of experiments compares LDA to other recommendation approaches and compares how the approximate inference choice affects the LDA's performance. The second experiment investigates whether the LDA is really personalising its recommendations to users. The third and final experiment compares LDA to the other recommendation approaches in a cold-start scenario.

The other recommendation strategies were chosen as the approaches that have performed best in other experiments looking at top-N recommendation. Some unpersonalised baselines were also included to provide a lower bound that a recommendation system should beat.

6.1 Evaluation Criteria

The recommender systems are evaluated for top-N recommendation where each recommender presents its top-N predicted items. Methods and models that predict ratings will first predict all the ratings for all user-item combinations and then give the top-N with the highest predicted ratings. LDA generates the top-N list by returning the items with the highest probability.

6.1.1 Choice of N

For the number of items to be presented by the recommender systems, the value of $N = 20$ was chosen to match the observation from Netflix that if a user does not find an item within 10-20 items they will leave (Gomez-Uribe & Hunt, 2015).

6.1.2 Precision and Recall

The approaches are evaluated with the accuracy metrics: precision, recall and the composite metric $F1$.

$$precision = \frac{\text{Total number of relevant items retrieved}}{\text{Total number of items retrieved}} \tag{6.1}$$

The highest precision achievable by any approach is 1 if every retrieved item in the top-N is relevant.

$$recall = \frac{\text{Total number of relevant items retrieved}}{\text{Total number of relevant items}} \tag{6.2}$$

The upper bounded of recall is $\frac{N}{\text{Number of relevant items}}$.

$$F1 = 2\frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{6.3}$$

To calculate the metrics, some of the user's ratings will be held out in a test set. An item is considered relevant in the top-N if it is in the test set.

6.2 The MovieLens 1 Million Ratings Dataset

The dataset used is the MovieLens 1 Million Ratings. The dataset is a collection of 1 million ratings from 6000 users on 4000 movies released in February 2003. With 6000 users and 4000 movies the feedback matrix has 24,000,000 possible user-item pairs. With 1 million ratings observed, this gives a sparsity of about 0.04%. The ratings are integer ratings on a scale between 1 and 5.

6.2.1 Conversion to Implicit Dataset

The Movielens dataset contains explicit ratings data, but the LDA model requires implicit count data. The explicit data can be converted to implicit data in several different ways.

The simplest is to have an implicit action be 'user u rated item i' so that $y_{u,i} = 1$ if $r_{u,i} \neq \emptyset$ and 0 otherwise.

Another method is to only note positive ratings. The implicit observation is now 'user u rated item i highly'. To do so we can compare the ratings to some threshold value that filters out negative ratings. The method then becomes: for each observed rating $r_{u,i}$ an implicit observation of the form 'the user u rated item i positively'. $y_{u,i}$ was generated by setting $y_{u,i}$ to 1 when $r_{u,i} \geq$ threshold and 0 otherwise. The threshold can be relative to each user, e.g. the user's average rating or static over all users. For these experiments we will use a threshold value of 3 for all users.

6.3 Recommender Systems That Were Compared

To compare LDA against traditional recommender models, we examined their performance for Top-N recommendation on the Movielens 1 Million Ratings dataset. In total, seven approaches are compared against one another: two baseline recommenders, a basic neighbourhood method, a PureSVD recommender, and an LDA recommender with inference performed with 3 different inference algorithms.

1. Baselines:
    a) Most Popular: Recommend items that have been rated the most.
    b) Highest Rated: Recommend items with the highest average rating.
2. Neighbourhood method.
3. PureSVD.

4. Latent Dirichlet Allocation:

    a) Online Variational Inference.

    b) Collapsed Gibbs Sampling.

    c) Expectation Propagation.

There are many possible variations for the core collaborative filtering approaches. The versions chosen here are mostly based on the best-performing models and methods for top-N recommendation. Cremonesi *et al.* (2010) compared the state-of-the-art matrix factorisation models, Asymmetric SVD and SVD++, against several other recommenders. Evaluated for top-N performance they found that PureSVD and a neighbourhood method performed best (recalls of 0.52 and 0.44 at N=10 in their setup).

6.3.1 Baseline Methods

Two unpersonalised recommender methods were chosen to provide a baseline against which to compare the other approaches. They provide the same recommendations to all users. We should expect any personalised recommender to beat these naive approaches.

The first baseline method recommends items that are rated the most, i.e. they are the most popular. The second recommends items with the highest average rating. Recommending the most popular items was included, as it has been shown to perform as well as some state-of-the-art matrix factorisation systems for top-N recommendation. Cremonesi *et al.* (2010) found that the most popular baseline approach got almost the same recall (0.28 versus 0.29 at N=10) as Asymmetric SVD. Asymmetric SVD is an evolution of the matrix factorisation model that came about during the Netflix Prize competition. Approaches that used Asymmetric SVD performed very well over the course of the competition.

6.3.2 Neighbourhood Method

For the neighbourhood method we will use cosine similarity as in Cremonesi *et al.* (2010). The ratings were pre-processed by normalising around the user

means.

6.3.3 Matrix Factorisation

PureSVD was used for the matrix factorisation approach. PureSVD was the best-performing matrix factorisation model for top-N in Cremonesi *et al.* (2010)'s results. Missing values in the feedback matrix were filled with 0 and ratings were normalised around user means.

6.3.4 LDA

For the LDA recommender, three different approximate inference algorithms were used to train the model. Both priors in the model were symmetric. The user prior was symmetrical, with the parameter being $\alpha_k = \frac{1}{K}$ for all k. The aspect prior was set to $\eta_i = 0.02$ for all i. $\boldsymbol{\eta}$ was chosen to not be sparse in an attempt to smooth the aspect item distributions and prevent them overfitting to the most popular items.

For the inference algorithms, VI was implemented following the online VI algorithm from Hoffman *et al.* (2010), which uses the same update equations as the original VI derivation from (Blei *et al.*, 2003). The Gibbs sampler was implemented as a collapsed Gibbs sampler as derived by Porteous *et al.* (2008). Finally the EP was implemented following Minka & Lafferty (2002).

6.4 Experiment 1: LDA for Recommendation

6.4.1 Methodology

The methodology was designed to simulate how recommenders are used in practice, where the system has a user's history and wishes to predict future items. The testing data was generated by splitting off the most recent twenty percent of each user's ratings into a test set. The training data then consisted of the first eighty percent of each user's ratings.

The training data was preprocessed for each model. The data was normalised

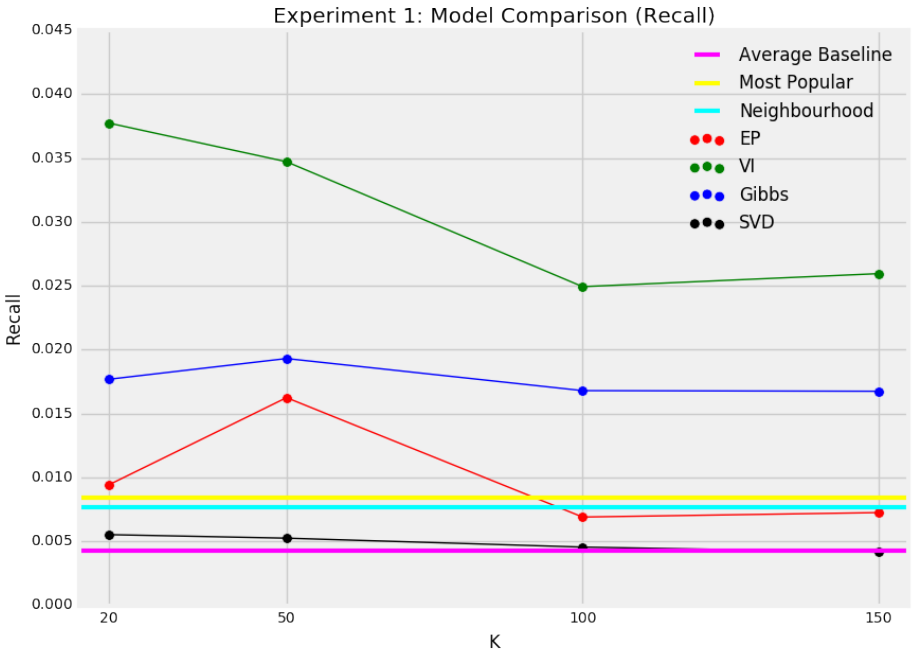
around the user means for the neighbourhood and PureSVD models.

For the LDA model the explicit ratings data were converted to implicit data with a threshold of 3.

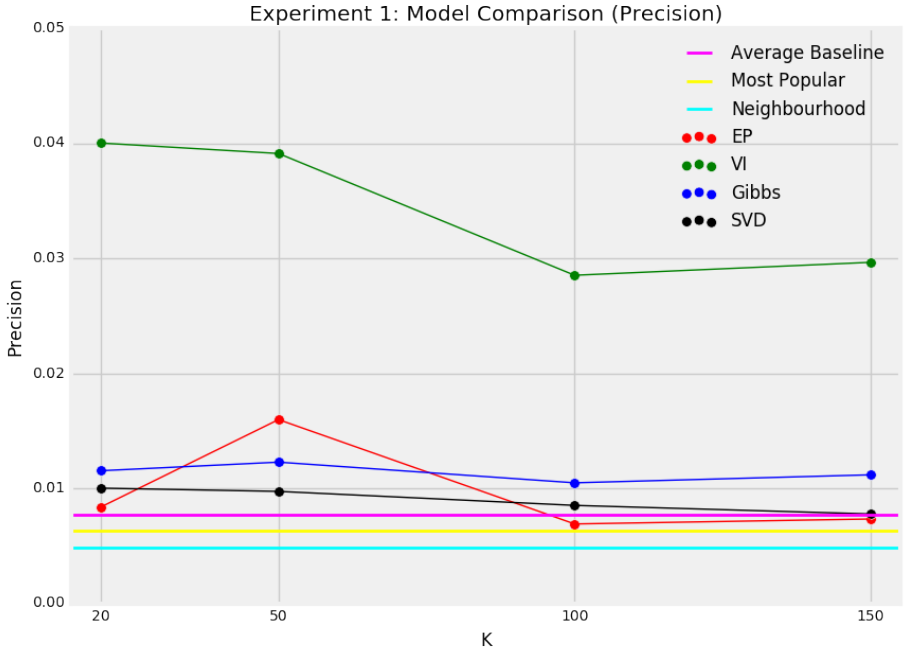
For PureSVD and LDA, the models were tested for four different numbers of features/aspects $K \in \{20, 50, 100, 150\}$ to examine performance at different ranks.

6.4.2 Results

The recall results are shown in figure 6.1a and the precision results in figure 6.1b. Focussing on precision LDA generally outperformed the other approaches. This is not surprising and is in agreement with the literature. The LDA model that used VI has a significant advantage over all the other approaches, but suffers mildly from inconsistent performance over the range of ranks of K. The PureSVD approach was consistent but only just outperformed the baselines and neighbourhood models. Gibbs sampling proved to be consistent regardless of the number of aspects. EP performed inconsistently and for some numbers of aspects extremely poorly, dipping below the average baseline approach.



(a) Experiment 1: Recall results (higher is better) when comparing LDA recommender against traditional recommender methods for top-N recommendation ($N = 20$). The LDA parameters were learned with three different approximate inference methods and the results are shown separately: VI, Gibbs sampling (Gibbs), and EP. VI and Gibbs for the LDA recommender performed well, and the LDA is effectively twice as good as the baseline approaches and traditional recommenders.



(b) Experiment 1: Precision results (higher is better) when comparing LDA against traditional and baseline recommender methods. With precision the LDA using VI performed far better than any other approach. The performance did drop off with higher numbers of aspects (K) where other approaches were constant over the range of features/aspects (where relevant).

To more clearly note the difference between the three approximate inference approaches, we can compare their F1 scores, as seen in figure 6.2.

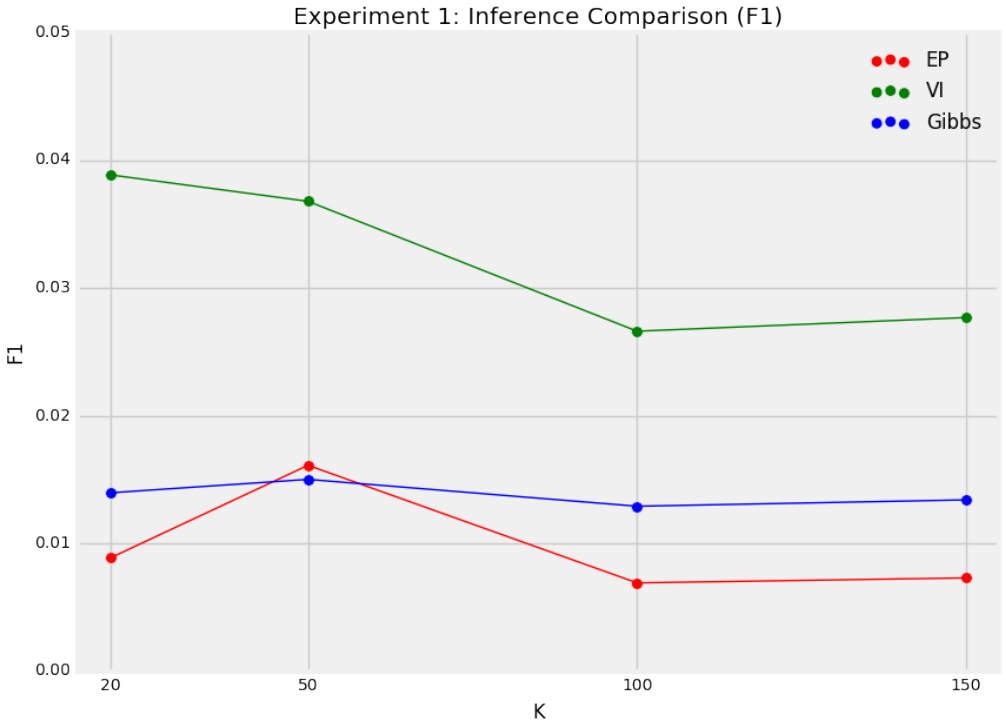


Figure 6.2: Experiment 1: Comparing the three different approaches to approximate inference via the F1 score for top-20 recommendation. Strong performance from VI that drops off with higher numbers of aspects. Gibbs is consistent across the range of aspects and EP lags behind the other approaches.

Here we can see the large performance gap between VI and the other approaches. The three approaches suggest a ranking of VI, then Gibbs, with EP as the worst approach for the LDA recommender. The difference in performance between VI and EP can possibly be explained by how EP averages over solutions. Averaging over good mixtures may not be a good mixture, where as VI is more likely to pick one of the possible mixtures.

6.5 Experiment 2: Effect of Personalisation

To try to understand how the variational LDA holds such an advantage over the other inference methods, we can examine how personalised the recommendations are. To do so we consider the difference in precision between the normal LDA recommendation and an unpersonalised LDA recommendation that simply recommends the items with the highest average probability in β. Another way of viewing this recommendation is to assume that all users have a uniform θ. By comparing the personalised and unpersonalised recommendation we can separate the performance of the LDA into the effect of finding popular items and increasing their likelihood in the aspect distributions in β, and having better partitioned aspects and personalising the users via their aspect mixtures θ.

The LDA performed best with 20 aspects and showed the largest discrepancy between VI and the others, so we consider $K = 20$. The recall scores for top-20 recommendation with 20 aspects are shown for the personalised and popular versions in figure 6.4

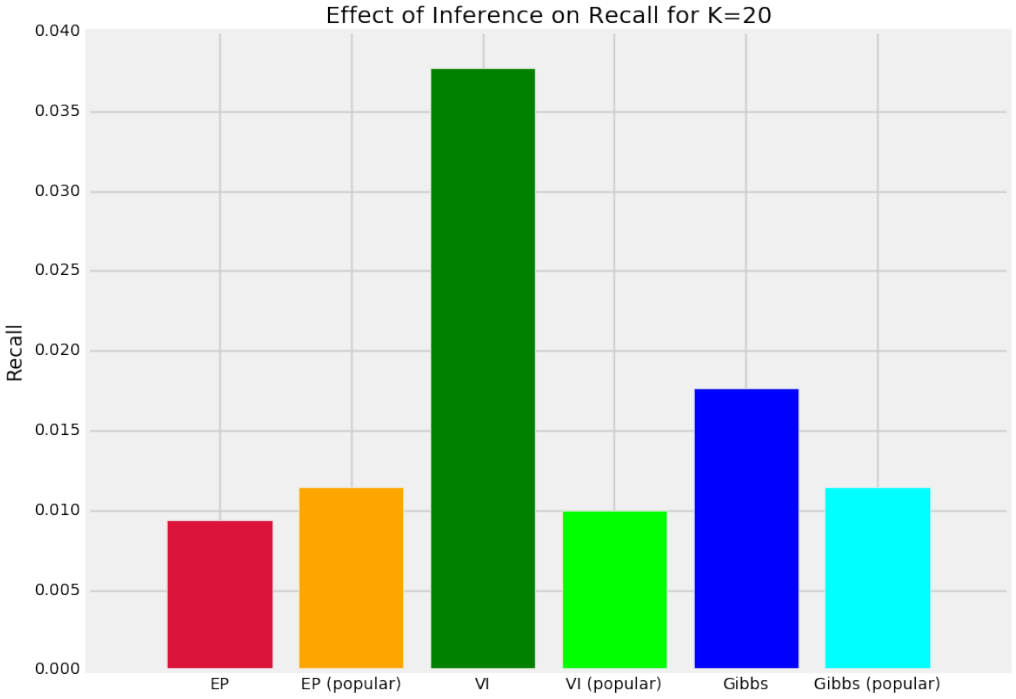


Figure 6.3: Experiment 2: Exploring how the LDA personalises results. The three approximate inference approaches are compared via their recall results with 20 aspects. The results with (popular) are unpersonalised, having a uniform θ for each user. All three approaches have roughly equal results for the popular recommendations, showing that VI and Gibbs get most of their accuracy improvements from better customisation to users rather than better average recommendation. More personalised recommendations is a good thing.

and the precisions are shown in figure 6.4.

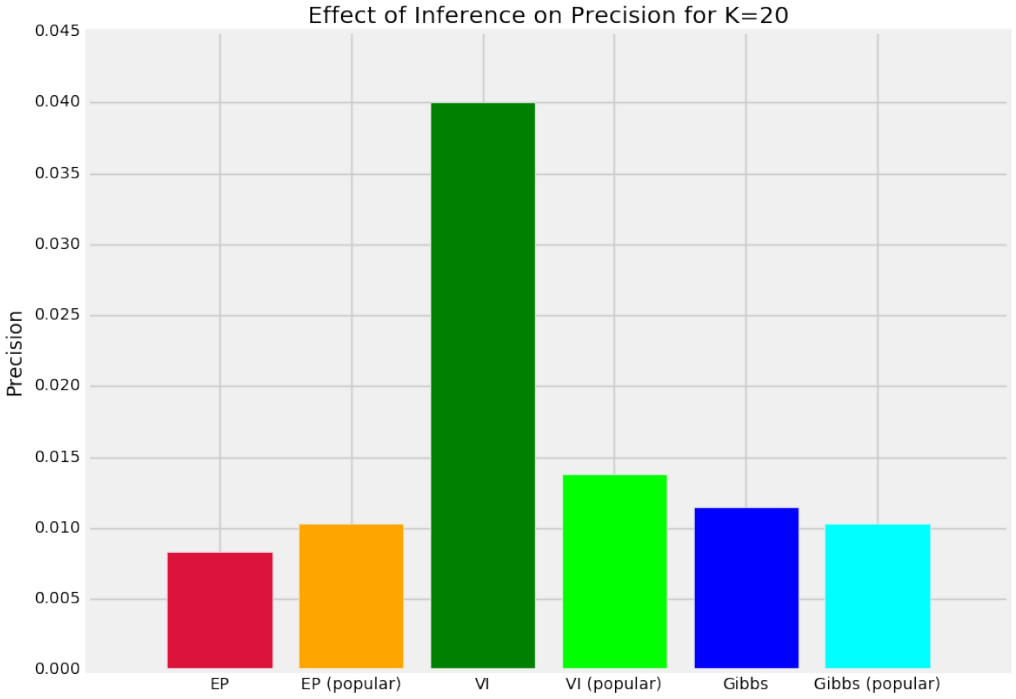


Figure 6.4: Experiment 2: Precision results. The precision shows a similar pattern to that seen with recall.

The unpersonalised versions of LDA all show relatively consistent performance for precision and recall. They all perform about as well as PureSVD (recall of 0.01). VI for LDA does not have significantly better results when only using β, suggesting that most of the gains are from better partitioning of users over aspects, not from better aspects. Only EP performed worse when the user mixtures were included. Note that the unpersonalised EP performed about as well as the other approaches (recall around 0.01) suggesting that the aspects are sensible but the assignment of users over them is the problem.

6.6 Experiment 3: Cold Start

A final experiment was performed to explore whether Latent Dirichlet Allocation performed well in cold-start scenarios. A test set was constructed by selecting twenty percent of users and leaving the first ten percent of their rat-

ings in the training set. The last ninety percent of their ratings constituted the test set. We just considered LDA with inference performed via VI, and compared it against the neighbourhood and PureSVD approaches. All model settings were the same as the first experiment. The precision results are shown in figure 6.5.

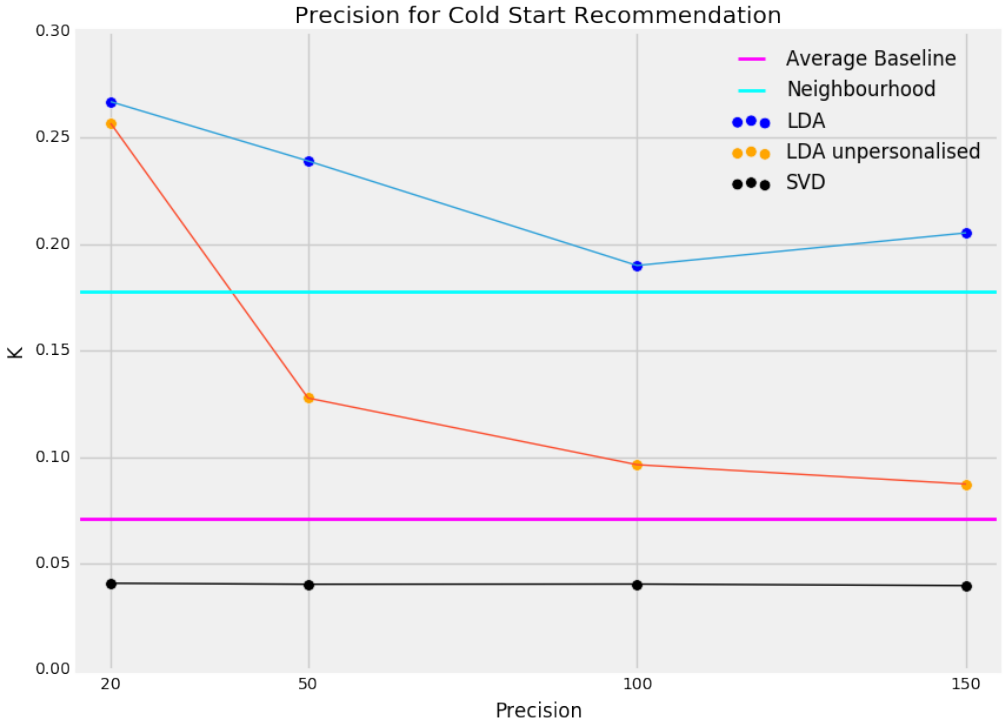


Figure 6.5: Experiment 3: Precision results for cold-start recommendation. LDA inference was performed with VI as it is the best performing. The personalised LDA still performs the best but neighbourhood methods also have perform similarly.

Latent Dirichlet Allocation once again performs the best. What is interesting to note is that for $K = 20$ the unpersonalised LDA performs almost as well as the LDA, but drops off for higher numbers of aspects. This is the opposite result to the first experiment where the VI LDA had a large discrepancy between the personalised and unpersonalised results. This may be due to the low number of aspects causing the aspect-item distributions to become more general. This in turn may cause the naive averages from the aspects to

include more popular items. The neighbourhood recommender also performs well as opposed to before, where it was one of the worst approaches. PureSVD performed below the baseline and is the worst recommender in this cold-start scenario.

6.7 Discussion

The first thing to note is that the results are consistent with the literature in that the traditional approaches perform poorly on top-N, sometimes even worse than the completely unpersonalised baselines. In general the probabilistic approach performed consistently better than the alternatives, further reinforcing the views of Barbieri & Manco (2011) that probabilistic approaches are preferred for item prediction. The implication for LDA is even better when we consider that the best-case performance with extensive hyperparameter tuning may be even better.

The performance split with different inference algorithms is stark. EP performed near the baselines where VI performed by far the best across all results. Experiment 2, which explored whether the LDA was personalising, gave some insight into why this discrepancy possibly exists. Bishop (2006) has suggested that EP tends to perform worse on mixture models than VI. The LDA model has a strong relationship with mixture models. Looking at the objective for EP $KL(p||q)$ may explain why, specifically the tendency to average over solutions. The average over two good aspect mixture assignments is probably not a good assignment. This is backed up by the fact that removing the effect of θ actually improves the EP results.

Turning to cold-start LDA once again produced convincing results and the equivalent competitor, SVD, fell to worse than baseline levels. The LDA results were also strongly personalised despite how little data was available. This highlights an advantage of generative models in general, i.e. the ability to quickly start drawing reasonable conclusions. The results do not jump around because of the damping effect of the prior, but instead slowly move towards sensible values. Empirical Bayes is another interesting avenue to explore that

may improve the results of LDA for cold start. In the case where $\boldsymbol{\alpha}$ is learned from data, a new user can be sampled from the prior

$$\boldsymbol{\theta}_{\text{new user}} \sim Dir(\boldsymbol{\alpha}) \tag{6.4}$$

to get an averaged starting point that reflects the relative popularity of the aspects, possibly improving the results for new users and essentially learning a better starting point for new users.

6.8 Summary

In conclusion, LDA is a strong contender as a starting point when building a modern recommender system, but the choice of inference algorithm cannot be ignored. Fortunately, VI has many advantages alongside the strong results shown here:

1. Automatic VI, ADVI, means that new models and extensions to LDA can be trialled without having to derive inference algorithms.

2. Online VI (SVI) can be derived from any CAVI algorithm, making inference efficient and extremely scalable.

3. With SVI, new users can be learned iteratively as they are streamed in without having to revisit the whole dataset.

Chapter 7

Conclusion

Latent Dirchlet Allocation has proven to be a flexible and powerful model in topic modelling. It highlights the best of Bayesian models, allowing for rich interpretation that can build quickly from limited or noisy datasets. The results when applying Latent Dirichlet Allocation to recommendation tasks are very promising. It consistently outperformed the popular collaborative filtering approaches and proved viable for the ever-challenging cold-start problem.

The drawback to using a Bayesian model is that we are reliant on approximate inference, the effect of which cannot be ignored. Depending on the inference approach chosen, Latent Dirchlet Allocation could have appeared to be a clear leader or a resoundingly average recommender. Of the three approximate inference algorithms we covered, Variational Inference proved to be the best for this application, but it has many other advantages. It is easy to derive from the Bayes net representation of the model, and with systems such as ADVI inference can even be derived automatically. It scales very well with the stochastic approach, simply using the same update equations to get online or parallel algorithms, and the SVI approach to LDA has been shown to scale to millions of documents when used for topic modelling.

To further expand Latent Dirichlet Allocation for recommender systems, relaxing some of the model assumptions or introducing extensions to the model could be considered. One assumption that the LDA makes is that the number of aspects is known. We can turn to Bayesian non-parametrics to address this.

Once again drawing from topic modelling, the non-parametric topic model determines the number of aspects during inference and new documents can display new unseen aspects(Teh *et al.*, 2005; Blei, 2012). This model can be extended even further to find hierarchies of aspects where a tree of topics is inferred from the data (Blei *et al.*, 2010; Blei, 2012). These non-parametric models could prove interesting for recommendation. In general, looking to the advances in topic modelling can bring inspiration for recommendation with these models. For example, advances such as *relational topic modelling* (Chang *&* Blei, 2010) could allow for recommender systems based on LDA that exploit links between users (social structure of the user base) - or to extend the LDA to incorporate metadata, instead of relying solely on feedback from users.

In closing, Latent Dirichlet Allocation is a promising approach for recommendation. In addition to improved accuracy over the previous popular collaborative filtering approaches, it is a rich and flexible model that can provide great insight into datasets. By drawing lessons from the field of topic modelling, many extensions and improvements to the LDA have already been developed.

Bibliography

Adomavicius, G. & Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on knowledge and data engineering*, 17(6):734–749.

Agarwal, D. & Chen, B.-C. 2010. flda: Matrix factorization through latent dirichlet allocation. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 91–100. New York, NY, USA: ACM. ISBN 978-1-60558-889-6.
Available at: `http://doi.acm.org/10.1145/1718487.1718499`

Allenby, G.M., Rossi, P.E. & McCulloch, R.E. 2005. Hierarchical bayes models: a practitioners guide.

Barbieri, N. & Manco, G. 2011. An analysis of probabilistic methods for top-n recommendation in collaborative filtering. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECML PKDD'11, pages 172–187. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-23779-9.

Bell, R.M. & Koren, Y. 2007. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79. ISSN 1931-0145.
Available at: `http://doi.acm.org/10.1145/1345448.1345465`

Bell, R.M., Koren, Y. & Volinsky, C. 2008. The Bellkor 2008 solution to the Netflix prize.
Available at: `http://signallake.com/innovation/Bellkor2008.pdf`

Bennett, J., Lanning, S. *et al.*. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, pages 3–6. Netflix, Los Gatos, CA 95032, USA.

Bernardo, J.M. 1996. The concept of exchangeability and its applications. *Far East Journal of Mathematical Sciences*, 4:111–122.

Bernardo, J.M. 2003. Bayesian statistics. *Encyclopedia of Life Support Systems (EOLSS). Probability and Statistics, (R. Viertl, ed).*
Available at: `http://www.uv.es/bernardo/BayesStat.pdf`

Bishop, C.M. 2006. *Pattern recognition and machine learning*. Springer.

Blei, D.M. 2005. Expectation propagation explanation.
Available at: `http://www.cs.columbia.edu/~blei/papers/Blei2003.pdf`

Blei, D.M. 2011. Variational inference. Lecture from Princeton.
Available at: `https://www.cs.cprinceton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf`

Blei, D.M. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Blei, D.M., Griffiths, T.L. *&* Jordan, M.I. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7.

Blei, D.M., Kucukelbir, A. *&* McAuliffe, J.D. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Blei, D.M. *&* Lafferty, J.D. 2009. Topic models. In A.N. Srivastava *&* M. Sahami (eds.), *Text Mining: Classification, Clustering, and Applications*, chapter 10, pages 71–94. Oxford: CRC Press.

Blei, D.M., Ng, A.Y. *&* Jordan, M.I. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Boyd, S. & Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

Breese, J.S., Heckerman, D. & Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1-55860-555-X.

Buntine, W.L. 1994. Operations for learning with graphical models. *J. Artif. Int. Res.*, 2(1):159–225. ISSN 1076-9757.

Burnham, K.P. & Anderson, D.R. 2003. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media.

Chang, J. & Blei, D.M. 2010. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, pages 124–150.

Charniak, E. 1991. Bayesian networks without tears. *AI magazine*, 12(4):50–63.

Chen, T., Zheng, Z., Lu, Q., Zhang, W. & Yu, Y. 2011. Feature-based matrix factorization. *ArXiv e-prints*, abs/1109.2271.
Available at: `http://arxiv.org/abs/1109.2271`

Clark, D.R. & Thayer, C.A. 2004. A primer on the exponential family of distributions. In *Casualty Actuarial Society Spring Forum*, pages 117–148.

Cremonesi, P., Koren, Y. & Turrin, R. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM.

Das, A.S., Datar, M., Garg, A. & Rajaram, S. 2007. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280. New York, NY, USA: ACM. ISBN 978-1-59593-654-7.
Available at: `http://doi.acm.org/10.1145/1242572.1242610`

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. *&* Sampath, D. 2010. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296. New York, NY, USA: ACM. ISBN 978-1-60558-906-0.

Davis, J. *&* Goadrich, M. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

Ekstrand, M.D., Riedl, J.T., Konstan, J.A. *et al.*. 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173.

Fink, D. 1997. A compendium of conjugate priors.
Available at: `https://www.johndcook.com/CompendiumOfConjugatePriors.pdf`

Fox, C.W. *&* Roberts, S.J. 2012. A tutorial on variational bayesian inference. *Artificial intelligence review*, pages 1–11.

Funk, S. 2006. Netflix update: Try this at home.
Available at: `http://sifter.org/simon/journal/20061211.html`

Funk, S. 2007. Netflix svd derivation.
Available at: `http://sifter.org/simon/journal/20070815.html`

Gelman, A., Vehtari, A., Jylänki, P., Robert, C., Chopin, N. *&* Cunningham, J.P. 2014. Expectation propagation as a way of life. *arXiv preprint arXiv:1412.4869*.

Geyer, C. 2017. Burn-in is unnecessary.
Available at: `http://users.stat.umn.edu/~geyer/mcmc/burn.html`

Gilks, W.R., Richardson, S. *&* Spiegelhalter, D. 1995. *Markov chain Monte Carlo in practice*. CRC press.

Golub, G.H. *&* Reinsch, C. 1970. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420.

Gomez-Uribe, C.A. *&* Hunt, N. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4):13:1–13:19. ISSN 2158-656X.
Available at: `http://doi.acm.org/10.1145/2843948`

Griffiths, T. 2002. Gibbs sampling in the generative model of latent dirichlet allocation.
Available at: `https://people.cs.umass.edu/~wallach/courses/s11/cmpsci791ss/readings/griffiths02gibbs.pdf`

Harper, F.M. *&* Konstan, J.A. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19. ISSN 2160-6455.

Herlocker, J.L., Konstan, J.A. *&* Riedl, J. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250. New York, NY, USA: ACM. ISBN 1-58113-222-0.
Available at: `http://doi.acm.org/10.1145/358916.358995`

Herlocker, J.L., Konstan, J.A., Terveen, L.G. *&* Riedl, J.T. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53. ISSN 1046-8188.

Hill, W., Stead, L., Rosenstein, M. *&* Furnas, G. 1995. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co.

Hoffman, M.D., Blei, D.M. *&* Bach, F. 2010. Online learning for latent dirichlet allocation. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pages 856–864. USA: Curran Associates Inc.
Available at: `http://dl.acm.org/citation.cfm?id=2997189.2997285`

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115. ISSN 1046-8188.
Available at: `http://doi.acm.org/10.1145/963770.963774`

Hofmann, T. *&* Puzicha, J. 1999. Latent class models for collaborative filtering. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 688–693. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hogg, R.V. *&* Craig, A.T. 1978. *Introduction to mathematical statistics.(4th edition)*. Macmillan Publishing Co., Inc.

Jaynes, E.T. 2003. *Probability theory: The logic of science*. Cambridge University Press.

Jin, R., Si, L. *&* Zhai, C. 2006. A study of mixture models for collaborative filtering. *Information Retrieval*, 9(3):357–382.

Klema, V. *&* Laub, A. 1980. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176.

Koller, D. *&* Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT Press.

Konstan, J.A., Riedl, J., Borchers, A. *&* Herlocker, J.L. 1998. Recommender systems: A grouplens perspective. In *Recommender Systems: Papers from the 1998 Workshop (AAAI Technical Report WS-98-08)*, pages 60–64.

Koren, Y. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Koren, Y., Bell, R. *&* Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):42–49.

Kucukelbir, A., Ranganath, R., Gelman, A. *&* Blei, D. 2015. Automatic variational inference in stan. In *Advances in neural information processing systems*, pages 568–576.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A. *&* Blei, D.M. 2016. Automatic differentiation variational inference. *arXiv preprint arXiv:1603.00788*.

Kullback, S. 1959. *Information Theory and Statistics*. Dover Publications, Inc.

Kullback, S. *&* Leibler, R.A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lam, X.N., Vu, T., Le, T.D. *&* Duong, A.D. 2008. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211. ACM.

Linden, G., Smith, B. *&* York, J. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80.

Marlin, B.M. 2004. Modeling user rating profiles for collaborative filtering. In *Advances in neural information processing systems*, pages 627–634.

McNee, S.M., Riedl, J. *&* Konstan, J.A. 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101. New York, NY, USA: ACM. ISBN 1-59593-298-4.

Minka, T. 2000. Estimating a dirichlet distribution.

Minka, T. *&* Lafferty, J. 2002. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc.

Minka, T. *et al.*. 2005. Divergence measures and message passing. Technical Report, Technical report, Microsoft Research.

Minka, T.P. 2001. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

Mnih, A. *&* Salakhutdinov, R.R. 2008. Probabilistic matrix factorization. In J.C. Platt, D. Koller, Y. Singer *&* S.T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pages 1257–1264. Curran Associates, Inc.
Available at: `http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf`

Ng, A.Y. *&* Jordan, M.I. 2002. On discriminative versus generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

Opper, M. *&* Winther, O. 2000. Gaussian processes for classification: Mean-field algorithms. *Neural computation*, 12(11):2655–2684.

Opper, M. *&* Winther, O. 2005. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6(Dec):2177–2204.

Ovsjanikov, M. *&* Chen, Y. 2010. Topic modeling for personalized recommendation of volatile items. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD'10, pages 483–498. Berlin, Heidelberg: Springer-Verlag. ISBN 3-642-15882-X, 978-3-642-15882-7.
Available at: `http://dl.acm.org/citation.cfm?id=1888305.1888337`

Pearl, J. *&* Russel, S. 2001. Bayesian networks. *Handbook of Brain Theory and Neural Networks*.

Peebles, P.Z. *&* Shi, B.E. 2001. *Probability, random variables, and random signal principles*, volume 3. McGraw-Hill New York, NY.

Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P. *&* Welling, M. 2008. Fast collapsed Gibbs sampling for Latent Dirichlet Allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM.

Ramezani, M., Bergman, L., Thompson, R., Burke, R. *&* Mobasher, B. 2008. Selecting and applying recommendation technology. In *International Workshop on Recommendation and Collaboration in Conjunction with 2008 In-*

ternational ACM Conference on Intelligent User Interfaces, IUI, pages 613–620.

Reed, C. 2012. Latent dirichlet allocation: Towards a deeper understanding. Available at: `http://obphio.us/pdfs/lda\_tutorial.pdf`

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. *&* Riedl, J. 1994. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

Rouder, J.N., Morey, R.D. *&* Pratte, M.S. 2013. Hierarchical bayesian models. *Practice*, 1(5):10.

Rubin, T. *&* Steyvers, M. 2009. A topic model for movie choices and ratings. In *Proceedings of the Ninth International Conference on Cognitive Modeling*. Manchester, UK.

Schafer, J.B., Frankowski, D., Herlocker, J. *&* Sen, S. 2007. *Collaborative Filtering Recommender Systems*, pages 291–324. Berlin, Heidelberg: Springer Berlin Heidelberg.

Schein, A.I., Popescul, A., Ungar, L.H. *&* Pennock, D.M. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.

Shardanand, U. *&* Maes, P. 1995. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co.

Shlens, J. 2014. Notes on kullback-leibler divergence and likelihood. *ArXiv e-prints*, abs/1404.2000.
Available at: `http://arxiv.org/abs/1404.2000`

Smith, B. *&* Linden, G. 2017. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18.

Spangher, A. 2015. Builåding the next new york times recommendation engine. Online; Accessed: 2016.
Available at: `https://open.blogs.nytimes.com/2015/08/11/building-the-next-new-york-times-recommendation-engine/`

Teh, Y.W., Jordan, M.I., Beal, M.J. *&* Blei, D.M. 2005. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392.

Teh, Y.W., Newman, D. *&* Welling, M. 2007. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1353–1360.

Ting, K.M. 2011. Precision and recall. In *Encyclopedia of machine learning*, pages 781–781. Springer.

Vargas, S. *&* Castells, P. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 109–116. New York, NY, USA: ACM. ISBN 978-1-4503-0683-6.

Wang, C. *&* Blei, D.M. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456. New York, NY, USA: ACM. ISBN 978-1-4503-0813-7.
Available at: `http://doi.acm.org/10.1145/2020408.2020480`

Wei, X. *&* Croft, W.B. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 178–185. New York, NY, USA: ACM. ISBN 1-59593-369-7.
Available at: `http://doi.acm.org/10.1145/1148170.1148204`

Weisstein, E.W. 2002. Singular value decomposition.
Available at: `http://mathworld.wolfram.com/SingularValueDecomposition.html`

Weisstein, E.W. 2003. Frobenius norm.
Available at: `http://mathworld.wolfram.com/FrobeniusNorm.html`

Yildirim, I. 2012. Bayesian inference: Gibbs sampling.
Available at: `http://www.mit.edu/~ilkery/papers/GibbsSampling.pdf`